Model Evaluation
Part 1: Experiments

Machine Learning
mlvu.github.io
Vrije Universiteit Amsterdam

---

offline machine learning: the basic recipe

Abstract (part of) your proble… task.
Classification, Regression, Clustering, Density estimation, Generative Modeling, Online learning, Reinforcement Learning, Structured Output Learning

next lecture: Prepare your data

Choose your **instances** and their **features**.
For supervised learning, choose a target.

Choose your **model class**.
Linear models, Decision Trees, kNN,

Search for a good model.
Usually, a model comes with its own search method. Sometimes multiple options are available.

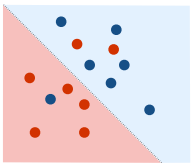
today: Evaluate your model

2

Here is the basic recipe for machine learning again. This week, we'll discuss what happens before and after. Today: once you've trained some models, how do you figure out which of them is best?

---

binary classification



Positive class

Negative class

The classifier is a detector for the positive class.

error: 3/14
proportion of misclassifications

accuracy: 11/14
proportion of correct classifications

3

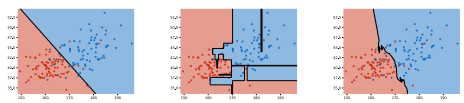
We'll focus mostly on binary classification today (two-class classification). In this case, we can think of the classifier as a detector for one of the classes (like spam, or a disease). We tend to call this class positive. As in "testing positive for a disease."

In classification, the main metric of performance is the proportion of misclassified examples (which we've already seen). This is called the **error**. The proportion of correctly classified examples is called the **accuracy**.

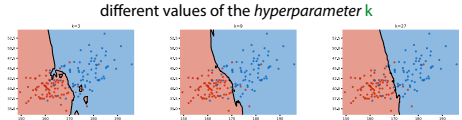
---

comparing models

linear vs. decision tree vs. kNN



different values of the *hyperparameter* k



4

You compare models to figure out which is the best. Ultimately, to choose which model to you want to use in *production*. (This could be literally the production version of a piece of software, or just the model whose predictions you decide to use in the future.)

Sometimes you are comparing different models types (decisions tree vs linear), but you might also be comparing different ways of configuring the same model type. For instance in the kNN classifier, how many neighbours (k) should we look at to determine our classification?

With the 2D dataset, we can look at the decision boundary, and make a visual judgment. Usually, that's

not the case: our feature space will have hundreds of dimensions, and we'll need to *measure* the performance of a model.

---

performing an experiment

Train classifier **A**, train classifier **B**

Compute the *error* of **A**, compute the *error* of **B**
error = proportion of mistakes

The lower the error, the better the model

On which data do we compute the error?

How do we eliminate random effects?

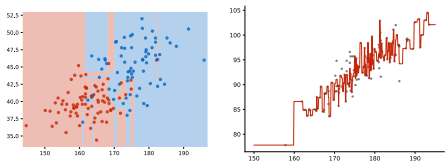
Is error the best metric to use?

5

Here is the simplest, most straightforward way to compare two classifiers. You just train them both, so see how many examples they get wrong, and pick the one that made fewest mistakes. This is a very simple approach, but it's basically what we do.

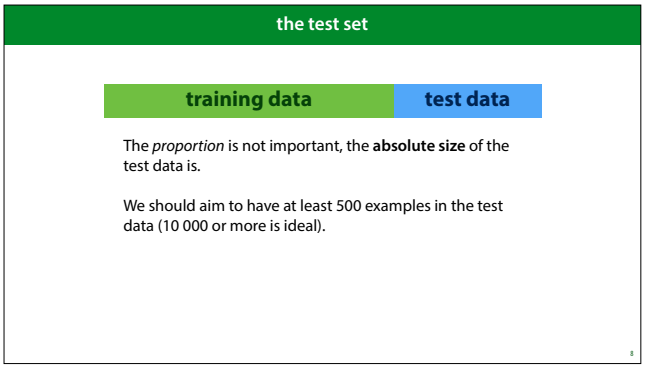
We just need to consider **a few questions**, to make sure that we can trust our results.

Overfitting



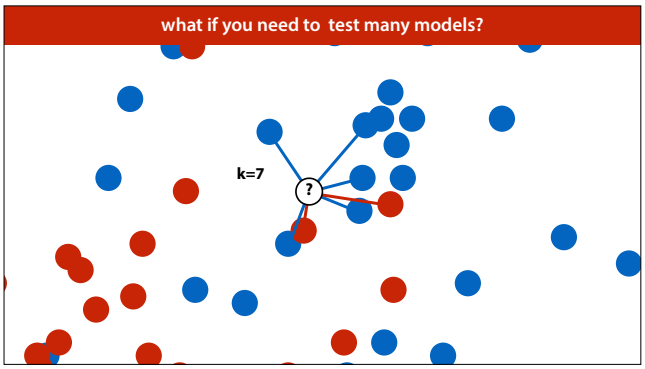
We've already seen what happens when you evaluate on the training data. A model that fits the training data perfectly may not be much use.

Never judge your performance on the training data

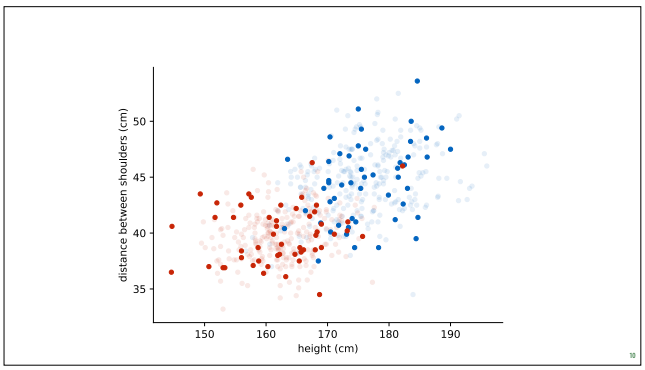


So the first thing we do in machine learning is withhold some data. We train our classifiers on the training data and test on the test data. That way, if we get good performance, we know that we're likely to get a good performance on future data as well, and we haven't just memorised random fluctuations in the training data.

How should we split our data? The most important factor is the size in instances of the test data. The bigger this number, the more precise our estimate of our model's error. Ideally, we separate 10 000 test instances, and use whatever we have left over as training data. Unfortunately, this is not always realistic. We'll look at this a little more in the second half.

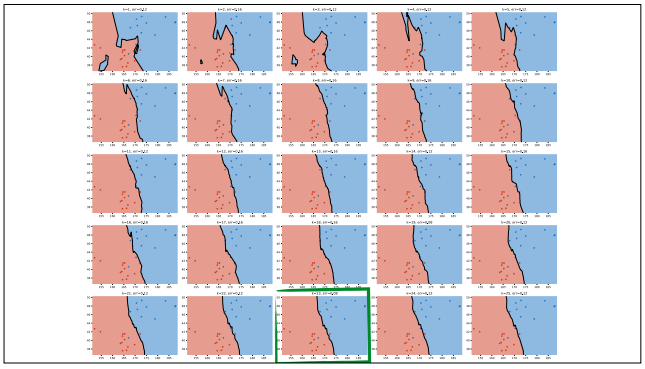


But even if we withhold some test data, we can still go wrong. We'll use k nearest neighbours as a running example. Remember, kNN assigns the class of the k nearest points.

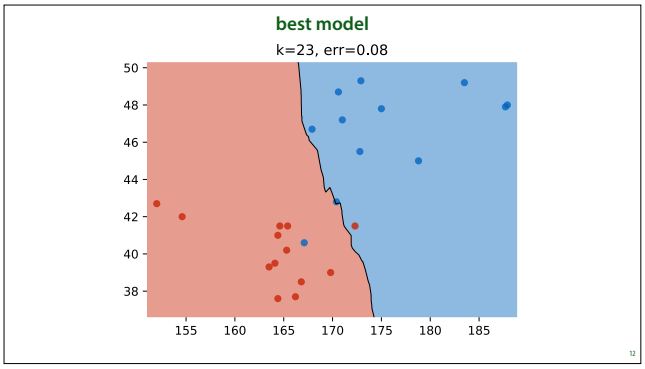


We will use the data from the first lecture as an example. We will take a small subsample, so that the effects become exaggerated

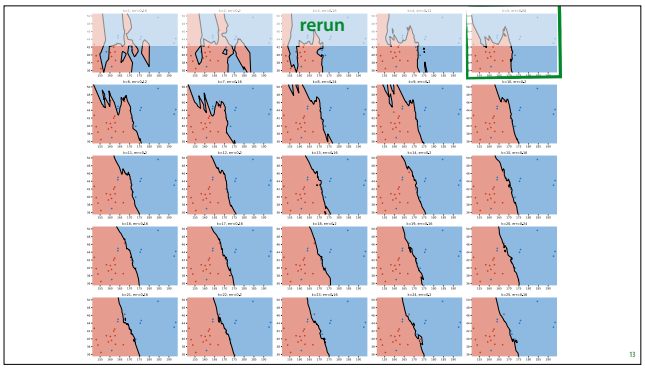
---



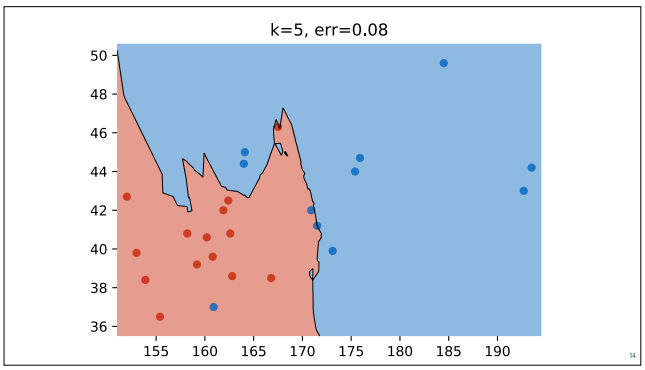
Here we've tested 25 different values of k on the same test data (using quite a small test set to illustrate the idea).

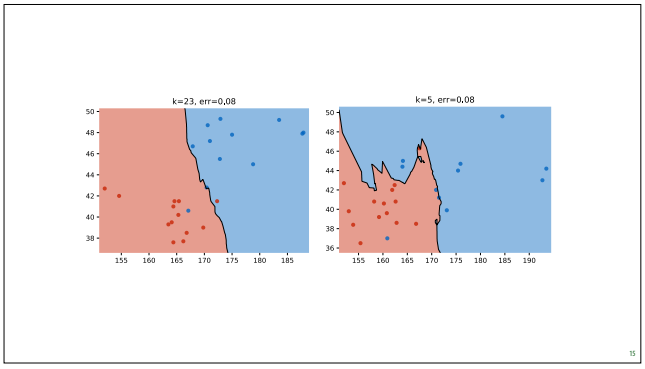


Here is the best run. Should we conclude that k=23 is definitely a better setting than k=22 or k=24? Should we conclude that we can expect an error of 0.08 on any future data from the same source?



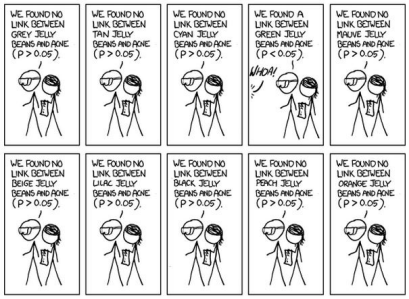
In this case, we have some more data from the same source, so we can do the whole experiment again in fresh data. Usually, this isn't the case, of course, and we use all the data we have.





The same source of data, the exact same procedure, and these are the results. Clearly, we can't trust that these models are actually learning the shape of the data.

conclusion: we're overfitting again



source: xkcd.com/882

16

This is essentially the overfitting problem again. Our method of choosing the hyperparameter **k** (or the model itself) is just another learning algorithm. By testing so many values of k on the test data, we are overfitting our choice of k on the test data.

This is an instance of the multiple testing problem in statistics. We're testing so many things, that the likelihood of a noticeable effect popping up by chance increases. We are in danger of ascribing meaning to random fluctuations. The simple answer to the problem of multiple testing is **not to test multiple times**.

see also: **https://www.explainxkcd.com/wiki/**

---

evaluation: the modern recipe

Split your data into train and test data.
Sample randomly. At least 500 examples in your test set. In ML benchmarks the test data is often given.

Choose your model, hyperparameters, etc. only using the training set.
Save your test set until the very last minute. Don't use it for anything.

State your hypothesis
i.e. *kNN with k=7 beats existing model X*, or *kNN with k=7 is better than kNN with k = 12*

Test your hypothesis **once** on the test data
This is usually at the very end of your project when you write your report or paper.

17

There are many different approaches to machine learning experimentation, and not every paper you see will follow this approach, but we believe this is the simplest way to get reliable results.

It's important to mention in your paper that you followed this approach, since the reader can't usually see it from the presented results.

Don't re-use your test data

Just to emphasize the important point: the more you use the test data, the less reliable your conclusions become. Figure out what the end of your project is, and do not touch the test data until the end.

In really important and long-term projects, it's not a bad idea to withhold multiple test sets. This allows you to still test your conclusions in case you've ended up using the original test data too often.

---

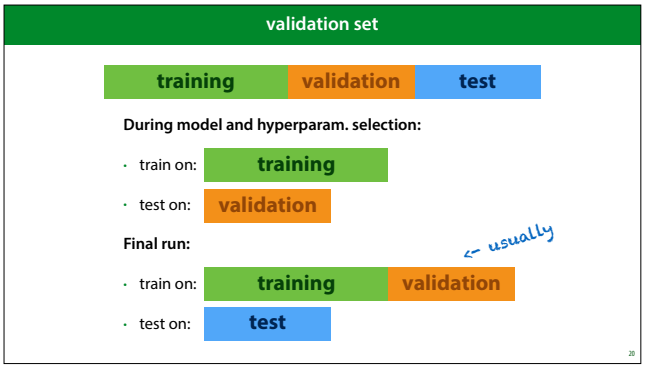
reusing your test data

Causes you to pick the wrong model

Inflates your performance estimate

19

Not only does reusing test data mean that you pick the wrong model, it also means that the error estimate you get is probably much lower that the error you would actually get if you gathered some more test data.

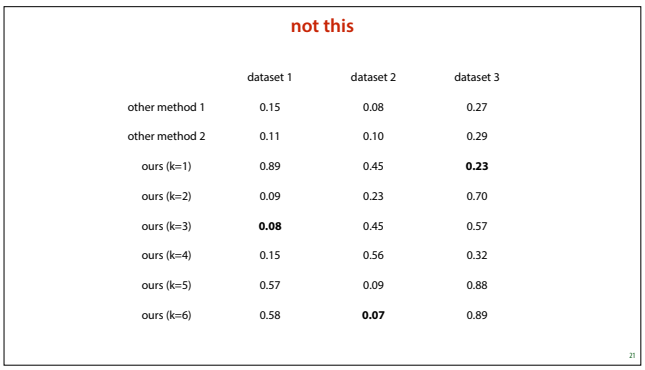


This means that you need to test which model to use, which hyperparameters to give it, and how to extract your features **only on the training data**. In order not to evaluate on the training data for these evaluations, you usually split the training data **again**: into a (new) **training set** and **a validation set**.

Ideally, your validation data is the same size as your test set, but you can make it a little smaller to get some more training data.

This means that you need to **carefully plan your research process**. If you start out with just a single split and keep testing on the same test data, there's no going back (you can't unsee your test data). And usually, you don't have the means to gather some new dataset.

It's usually fine in the final run to append the validation data to your training data. This is not always the case however, so if you use a standard benchmark you should check if this is allowed, and if you use new data you should describe carefully whether you do this.



This may seem like a simple principle, but it goes wrong **a lot**. Not just in student papers, also in published research.

Here's what you might come across in a bad machine learning paper. In this (fictional) example, the authors are introducing a new method (labeled *ours*) which has a hyperparameter k. They are claiming that their model beats every baseline, because their numbers are higher (for specific hyperparameters).

These numbers create three impressions that are not actually validated by this experiment:

- That the authors have a better model than the two other methods shown.
- That if you want to run the model on dataset 1, you should use k=3
- That if you have data like dataset 1, you can then expect an error of 0.08.

None of these conclusions can be drawn from this experiment, because we have not ruled out multiple testing.

but this

	dataset 1	dataset 2	dataset 3
other method 1	0.15	0.08	0.27
other method 2	**0.11**	0.10	0.29
k	3	5	2
ours	**0.11**	0.11	**0.24**

"The hyperparameter k was chosen based on a validation set split off from the training data. The test data was used only once."

22

Here is what we should do instead. We should use the training data to select our hyperparameters, make a single choice, and then estimate the accuracy of only that model.

Note that the numbers have changed, because in the previous example we gave ourselves an advantage by multiple testing. These numbers are lower, but more accurate. (I made these numbers up, but this is the sort of thing you might see)

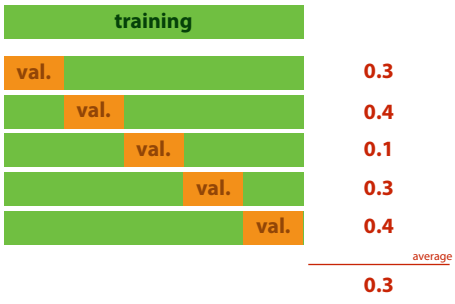
Now, we can actually draw the conclusions that the table implies:

- On dataset 3, the new method is the best.
- If we want to use the method on dataset 3 (or similar data) we should use k=2
- If our data is similar to that of dataset 3, we could expect a performance around 0.24

Even though most people now use this approach, you should still mention exactly what you did in your report (so people don't assume you got it wrong).

---

cross-validation



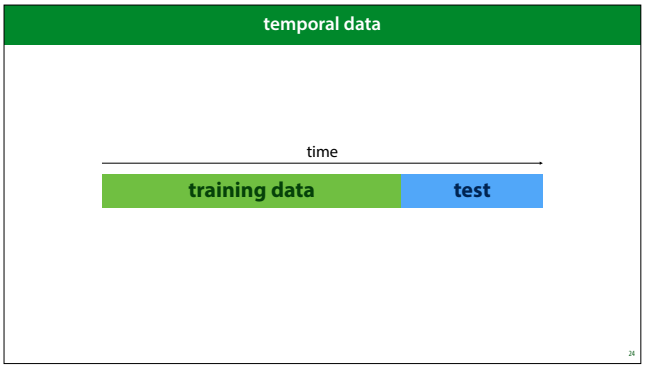
23

After you've split off a test and validation set, you may be left with very little training data. If this is the case, you can make better use of your training data by performing **cross-validation**. You split your data into 5 chunks ("folds") and for each specific choice of hyperparameters that you want to test, you do five runs: each with one of the folds as validation data. You then average the scores of these runs.

This can be costly (because you need to train five times as many classifiers), but you ensure that every instance has been used as a training example once.

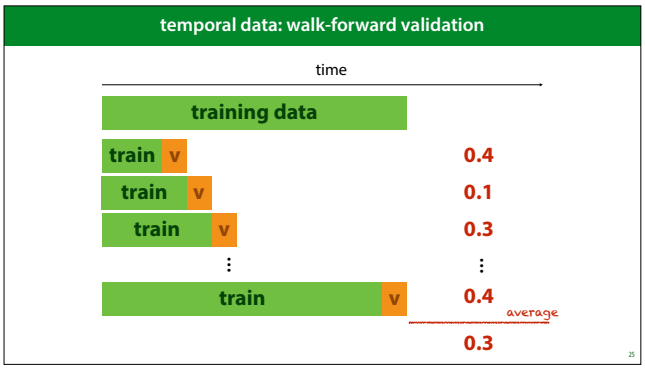
After selecting your hyperparameters with crossvalidation, you still test once on the test data.

You may occasionally see papers that estimate error of their finally chosen model by cross validation as well, but this is a complicated business, and has fallen out of fashion. We won't go into in this course.



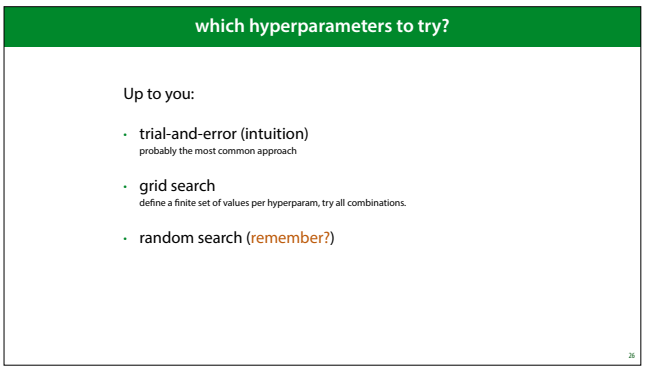
If your data has special attributes, like a meaningful temporal ordering of the attributes, you need to take this into account. In the case of temporal data, training on samples that are in the future compared to the test set is unrealistic, so you can't sample your test set randomly. You need to maintain the ordering.

Sometimes data has a timestamp, but there's no meaningful information in the ordering (like in email classification, seeing emails from the future doesn't give you an unfair advantage in the task). In such cases, you can just sample the test set randomly.



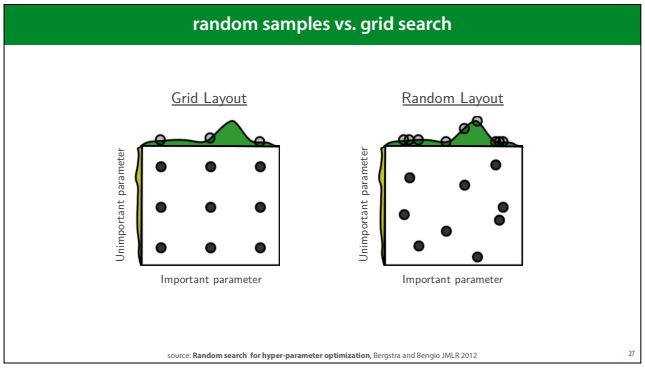
If you want to do cross-validation, you'll have to slice the dataset like this.

In general, don't just apply split testing and cross validation blindly, think about how you will ultimate train and use your model "in production". Make sure the validation setting is an accurate simulation of that setting.



Which values should we try for the hyperparameters? So long as we make sure not to look at our test set, we can do what we like. We can try a few values, we can search a grid of values exhaustively, or we can even use methods like random search, or simulated annealing

It's important to mention: **trail and error is fine, and it's the approach that is most often used**. Often, it's the most effective, because you (hopefully) have an intuitive understanding of what your hyperparameters mean.



Often, random samples, in the hyper parameter space are better than a grid. This picture neatly illustrates why. If one parameter turns out not to be important, and another does, a grid search restricts us to only three samples over the important parameter, at the cost training nine different models.

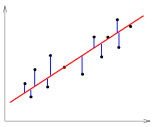
source: **http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf** (recommended reading)

Model Evaluation
Part 2: Evaluation

Machine Learning
mlvu.github.io
Vrije Universiteit Amsterdam

---

regression



loss function: (mean) squared errors

$$\frac{1}{n}\sum_i (f(x_i) - t_i)^2$$

evaluation function: root mean squared error

$$\sqrt{\frac{1}{n}\sum_i (f(x_i) - t_i)^2}$$

29

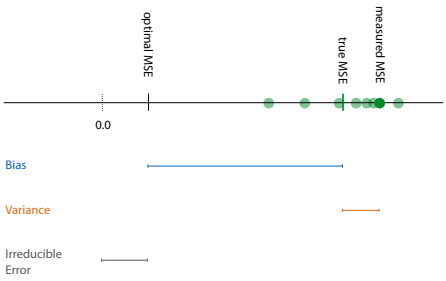
We'll quickly look at regression. We have much less to say here.

One thing to pay attention to is that if you use MSE loss, you may want to *report* the square root (the RMSE). The RMSE is minimised at the same places as the MSE, but it's easier to interpret, because it has the same units as the original output value.

For instance, if your outputs are in meters, then your MSE is measured in square meters, but your RMSE is also measured in meters.

---

bias and variance

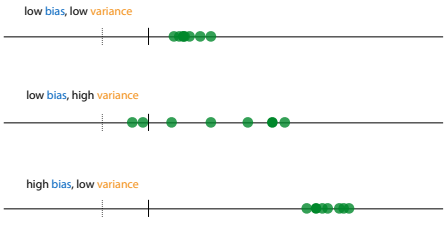


30

Normally, we train a regression model once, and get once MSE value. The lower the better. However this MSE is an estimate of the "true MSE" This is a value we can't compute, its the true expected performance of our entire method: from sampling data to training the model to computing the performance. If

---

bias and variance



31

bias and variance

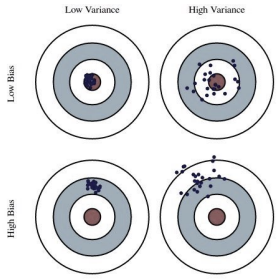


image source: http://scott.fortmann-roe.com/docs/BiasVariance.html (recommended reading)

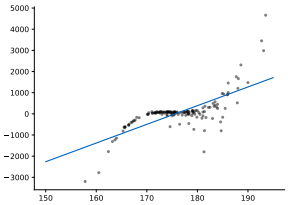
32

Remember, this is a metaphor for our RMSE error estimate. That means that normally, we have only one dart and we can't tell whether our error is due to high bias or high variance. We'll return to this topic in week 5, in the context of ensemble methods.

---

the bias-variance tradeoff

High bias: model doesn't fit the generating distribution. Poor assumptions, poor capacity. Aka *under*fitting.



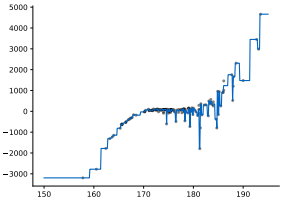
33

High bias tends to happen when the model can form the true shape of the data. Linear models in low-dimensional spaces often have this problem.

---

the bias-variance tradeoff

High variance: high model capacity, sensitivity to random fluctuations. aka *over*fitting



34

High variance happens when the model has the capacity to follow the shape of the data perfectly, but so perfectly that it tends to get thrown off by small fluctuations.

Even though this model (a regression tree) fits the data perfectly, if we resample the data, we are stuck with all sorts of weird peaks that won't fit the new data.

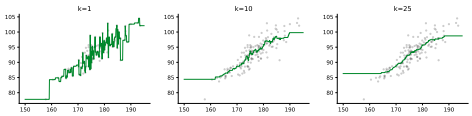
---

making the tradeoff

Reducing bias: increase model capacity, increase features.

Reducing variance: reduce model capacity, add regularization, reduce tree depth.

k-NN regression: increase k to increase bias, decrease variance.



35

We will see techniques for all of these in the coming weeks. Note that often, it really is a tradeoff: reducing the bias, increases the variance and vice versa.

For some algorithms, there is a single parameter that allows us to make the bias/variance tradeoff. kNN is one example.

lecture 10: ensembling

Combining models for variance reduction and for bias reduction.

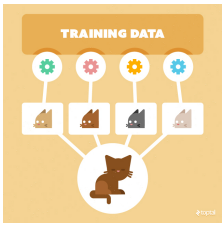


image source: https://www.toptal.com/machine-learning/ensemble-methods-machine-learning

16

In week

evaluating classification

Class imbalance

Confusion matrix

True positive rate, True negative rate
More in the next video

Precision, recall

17

Is an error of 0.05 good?

18

Now that we know how to properly estimate the classification error of our model, let's see what it means.

Imagine that somebody tells you about a machine learning project they're doing, and they proudly state that they get a classification error (on their validation set) of 0.05 (5% of the validation set is misclassified). Should you be impressed?

example: breast cancer screening

Redt preventieve screening op borstkanker levens?

Voor- en tegenstanders van de mammografie

ARTIKEL De ene wetenschapper beschouwt de massale screening op borstkanker als levensreddend, volgens de ander zitten er meer nadelen dan voordelen aan. Een medisch vakblad spreekt al van een mammografieoorlog. Wat doe je nu als 50-plusvrouw met de oproep voor de borstenbus?

Door: Ellen de Visser 4 oktober 2014, 03.02

AANBEVOLEN ARTIKEL

'Stop screening borstk 70+-vrouwen' 16 september 2014

'De in zichzelf gekeerd veranderde in een vrol hij zijn stem verloor' 10 februari 2018

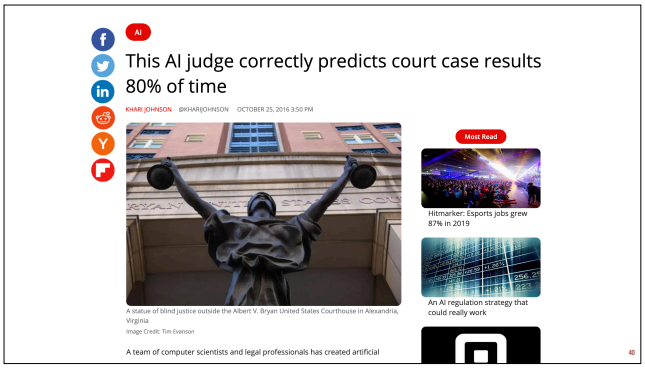
Miljarden euro's voor verpleeghuiszorg, maa waar het terechtkomt?

19

For an example, let's look at a recurring discussion in the Dutch media: should all women over 50 be screened for breast cancer. This is an analogy for classification: the instances are people and the target label is "has cancer" or "has no cancer."

If 1 percent of patients have cancer, an error of 0.05 is not very impressive. We can do much better by just saying that nobody has cancer.

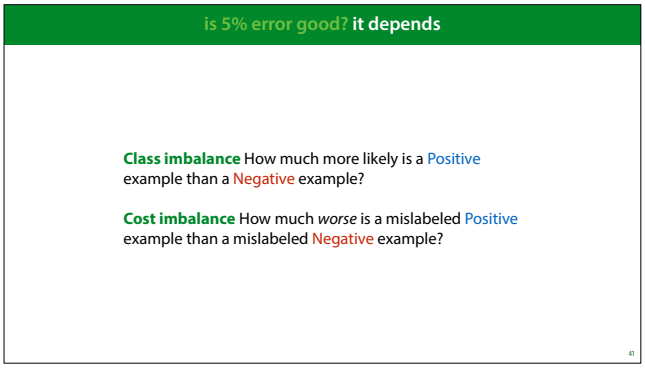
source: **https://www.volkskrant.nl/wetenschap/redt-preventieve-screening-op-borstkanker-levens~a3761451/**



So the next time you see a headline like this, your first question should be: what was the class distribution in the training data? If 90% of the cases in the training data are acquittals, this is not a very impressive result.

As it happens, in this case the classes were balanced 50/50, so 80 percent is at least notable. However, now we have a classifier *trained on artificially balanced data*. In a production environment (whatever that means here), the classes are likely *not* balanced 50/50, so this specific classifier will be of no further use.

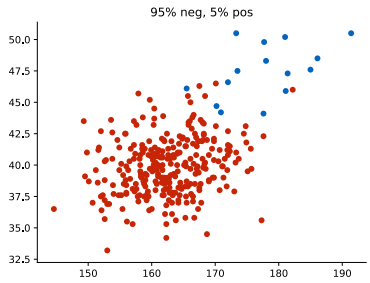
Here is the original paper: **https://peerj.com/articles/cs-93/#fn-6** There are some issues with this research beyond the class balance.



Imagine a classifier for breast cancer with 95% accuracy. This may seem impressive at first sight, but the prevalence of great cancer among women over 50 (the population that is regularly tested) is about 1%. This means that **a classifier that classifies nobody as having cancer** gets 99% accuracy. The range of accuracies that are interesting at all is between 99% and 100%.

Another reason to mistrust accuracy (besides class imbalance) is **cost imbalance**. There are two types of misclassification: diagnosing a healthy person with cancer and diagnosing a person with cancer as healthy. Both come with a cost but not the same cost. Spam classification is another example: both misclassifications should be avoided, but having a genuine email land in your spam is much worse than having to delete a spam email from your inbox.

class imbalance



42

Here is a pretty imbalanced dataset (though still not as imbalanced as the cancer/not cancer problem). It looks pretty difficult. What would be a good performance on this task?

---

class imbalance

other method 1	0.15
other method 2	0.1
majority class	**0.05**
ours	**0.05**

Majority class classifier Assigns all instances the class that is most prevalent in your data.

Example of a **baseline**.

43

Something like an error of 0.05 might sound pretty good, but on an imbalanced dataset like this, there is a very simple classifier that gets that performance easily: the **majority class classifier**

The majority class classifier is an example of a **baseline**, a simple method that is not meant to be used as a real model, but that can help you calibrate the performance scores. In this case, it tells you that you're really only interested in in the range from 0 to 0.05. Any higher error than that is pretty useless.

---

class imbalance

44

Here is another way that class imbalance can screw things up for you. You might think you have a pretty decent amount of data with 1000 instances. However if you split off a test set of 1000 instances, you're left with just 50 instances of the positive class in your data. Practically, your final evaluation will just be a question of how many of these 50 positives you detect. This means that you can really only have 50 "levels of accuracy" that you can distinguish between.

You can make a bigger test set of course (and you probably should) but that leads to problems in your training data. Since you're essentially building a detector for positives, it doesn't help if you can only give it 100 examples of what a positive looks like.

In the next lecture, we'll look at some tricks we can use to boost performance on such imbalanced data.

cost imbalance

disease diagnosis Sending a sick person home vs applying invasive tests to a healthy person

spam classification Deleting a valid email vs showing the user a single spam email

detecting financial fraud Having an expert review a non-fraudulent transaction vs missing a fraud in progress

Domain-specific evaluation function: dollars lost, time lost, lives lost, etc.

45

The second thing we need to consider when interpreting errors is **cost imbalance**.

In all these cases, one misclassification one way costs much more than a misclassification the other way. But both cost *something*. The time of an expert reviewer is not free, even though five minutes of his time may be much cheaper than the cost of letting a single fraud go unchecked.

If you're lucky, both types of misclassification have the same unit, and you can turn your error (an estimate of the number of misclassifictions) into a domain specific evaluation function (like estimated dollars lost, or time saved). **You simply assign a cost to each type if misclassification, and multiply it by how often that misclassification occurs in the test set.**

If the units are not the same (money saved vs lives saved) making such a choice can seem very unethical. On the other hand, any classifier you decide to deploy will *implicity* make such a choice. Even if you decide not to use machine learning, the alternative (a doctor using their own judgement) is also a "classifier", with its own cost balance.

---

social impact

Cost imbalance is particularly important when we consider matters of social impact. If we predict a person's sex from their physical appearance perfectly, and we use that as a prediction for their gender, we have 99% accuracy.

However the 1% we then misclassify is precisely that part of the part of the population for which gender is a sensitive attribute. Just because our classifier has high accuracy, doesn't mean it can do no harm. In a large part because the mistakes it makes are not uniformly distributed. They are focused squarely on the vulnerable part of the population.

---

other performance metrics

Confusion matrix

Precision, recall

True positive rate, false positive rate

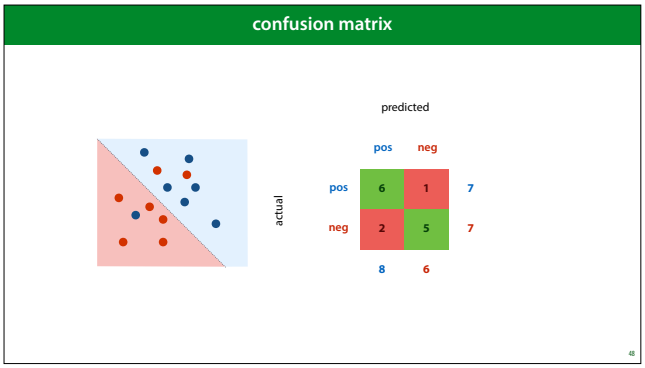
ROC plot, Coverage matrix, Area under the curve

Next video

47

The best thing to do under class and cost imbalance, is to look at your performance in more detail. We'll look at six different ways to measure classifier performance.

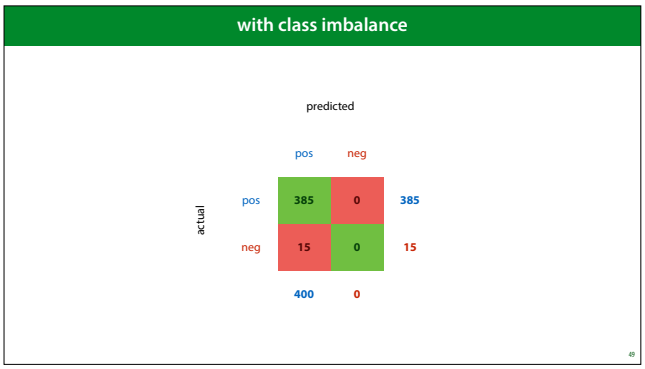
Most of these are only relevant if you have class or cost imbalance. If you have a nice, balanced dataset, it's likely that error or accuracy is all you need.



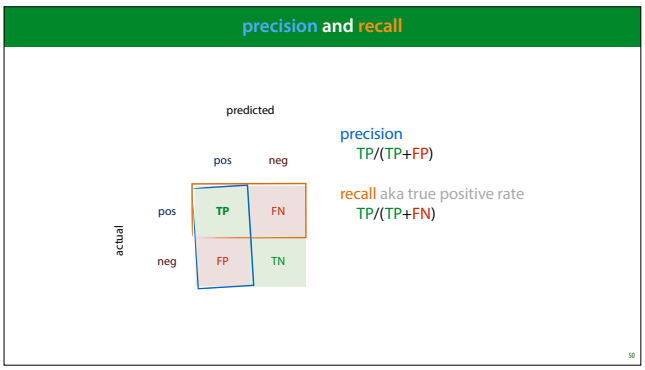
This is s a **confusion matrix** (also known as a contingency table). It doesn't give you a single number, so it's more difficult to compare two classifiers by their confusion matrices, but it's a good way to get insight into what your classifier is actually doing.

You can plot the confusion matrix for either the training, validation or test data. All three can be informative.

The margins of the table give us four totals: the actual number of each class present in the data, and the number of each class predicted by the classifier.



Here we see the confusion matrix for the majority vote baseline in a problem with high class imbalance.

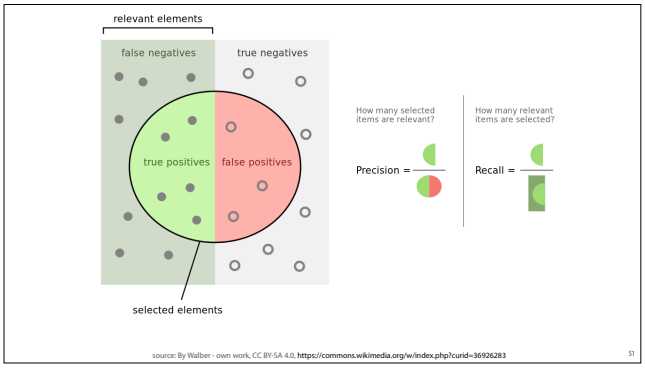


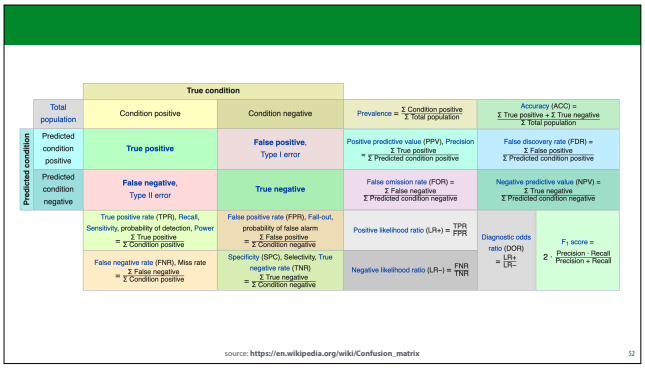
Precision and recall are two other metrics that express a similar tradeoff.

Precision: what proportion of the returned positives are actually positive?

Recall: what proportion of the existing positives did we find?

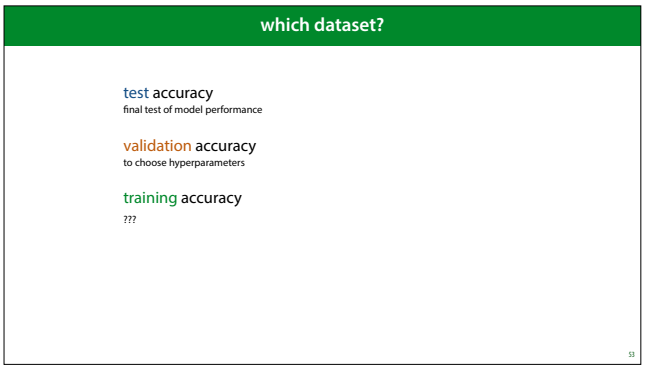
These can also be plotted in 2 axes. For now, we'll stick with the ROC space.





There are many more metrics which you can derive from the confusion matrix. Wikipedia provides a helpful table, in case you ever come across them. For most purposes, precision, recall accuracy and balanced accuracy are sufficient.

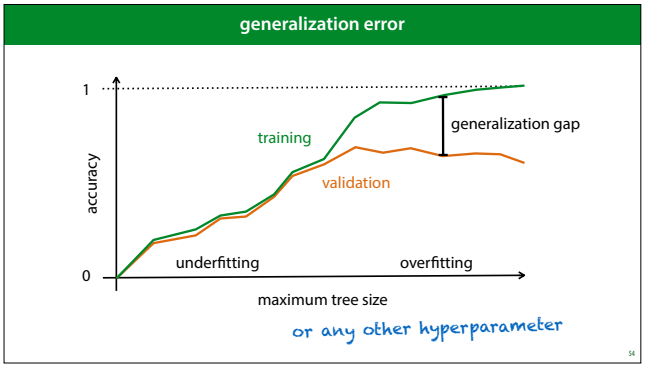
Note that some terms, like recall, go by *many* different names.



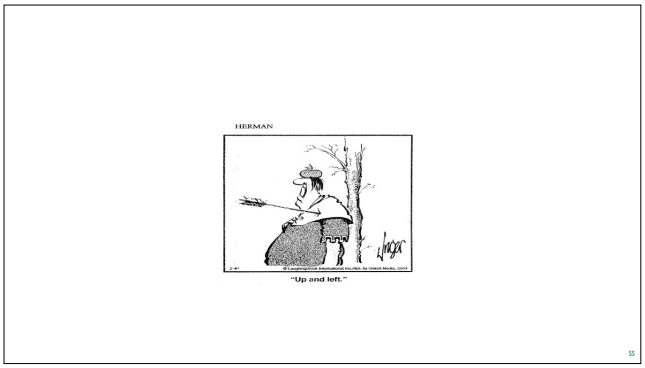
All of these metrics can be applied to different datasets. When we compute (say) accuracy on the test set, we talk about **test accuracy**. This is computed, only once, at the very end of our project, to show that our conclusions are true.

When we compute it on the validation set we call it **validation accuracy**. We compute this to help us choose good hyperparameters.

And, predictably, when we compute it on the training data, we call it **training accuracy**. Remember that in the first lecture I said, emphatically, that you should never judge your model on how it performs on the training set. Why then, would you ever want to compute the training accuracy (or any other metrics on the training data)?



The answer is that the difference between your validation error and your training error, will tell you whether of not your model is overfitting (matching the data too well) or underfitting (not matching the data well enough)



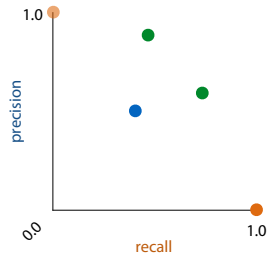
55

Model Evaluation
Part 3: PR, ROC and AUC

Machine Learning
mlvu.github.io
Vrije Universiteit Amsterdam

---

precision/recall space

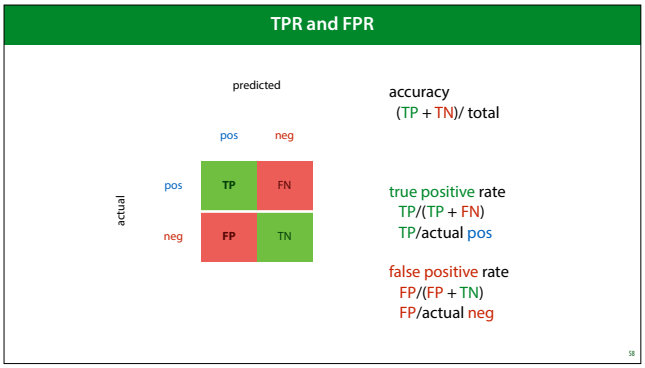


57

Clearly, we want to get both the precision and the recall as high as possible. Since we don't know how to balance them (this depends on domain specific preferences, i.e. cost imbalance), we can visualize both objectives together in the plane. Each classifier represents a point.

The points in the corners represent our most extreme options. We can easily get a 1.0 recall by calling everything positive (ensuring that all true positives are among the selected elements). We can get a very likely 1.0 precision by calling only the instance we're most sure about positive. If we're wrong we get a precision of 0, but if we're right we get 1.0.

Whether we prefer the left or the right green classifier depends on our preferences. However, whatever our preference, we always prefer either green classifier to the blue classifier.



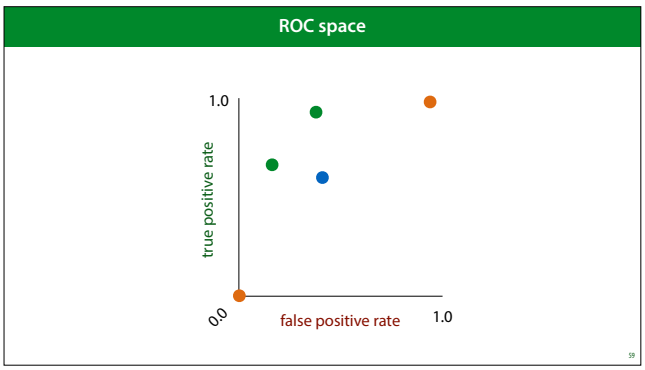
We call accurately classified instances **true positives** and **true negatives**. Misclassifications are called **false positives** and **false negatives**.

Many performance measures can be computed from the confusion matrix (including the error and accuracy).

A good pair of metrics to consider, especially when we have class imbalance, is the true and false positive rate:

true positive rate: what proportion of the actual positives did we get *right*. The higher the better. I.e. How many of the people with cancer did we detect.

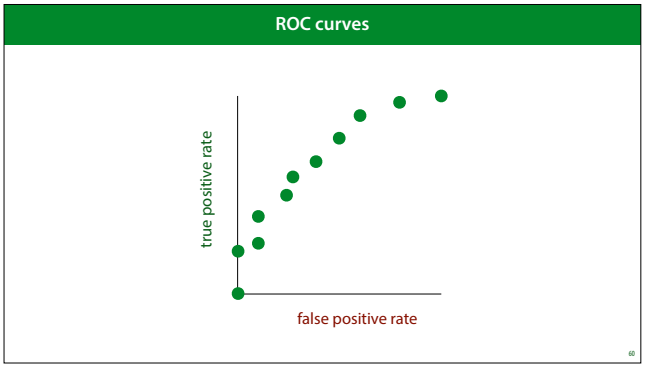
false positive rate: what proportion of the actual negatives did we get *wrong* (by labelling them as positives). The lower the better. I.e. How many healthy people did we diagnose with cancer.



We want to get the TPR as high as possible, and the FPR as low as possible. That means the TPR/FPR space has the best classifier in the top left corner. This space is called ROC space.

Again, the orange points are the extremes, and easy to achieve.

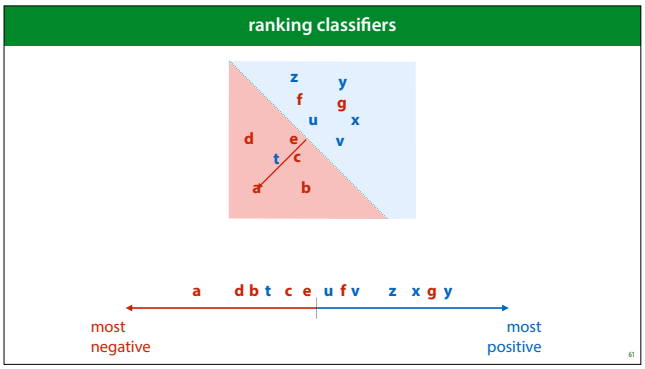
ROC stands for **receiver-operating characteristic**, a leftover from its invention in WWII, to improve the detection of Japanese aircraft from radar signals.



So far we've tough of FRP/TPR and precision/recall as a way to analyze a given set of models.

However, what if we had a *single* classifier, but we could control how eager it was to call things positive? If we made it entirely timid, it would classify nothing as positive and start in the bottom left corner. As it grew more brave, it would start classifying some things as positive, but only if it was really sure, and its true positive rate would go up. If we made it even more daring, it would start getting some things wrong and both the tpr and the fpr would increase. Finally, it would end up classifying everything as positive, and end up on the top right corner.

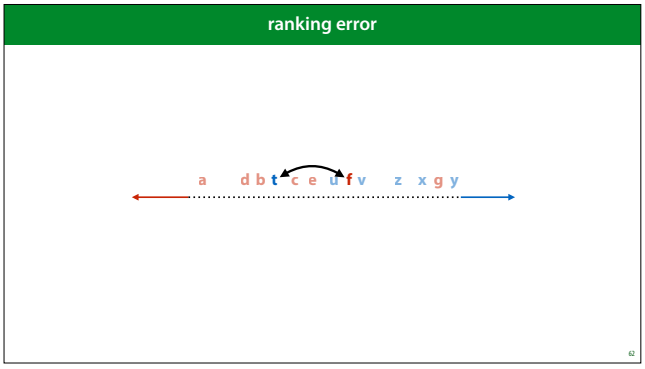
The curve this classifier would trace out, would give us an indication of its performance, *independent* of how brave or how timid we make it. How can we build such a classifier?



We can achieve this by turning a regular classifier into a **ranking classifier** (also known as a **scoring classifier**). A ranking classifier doesn't just provide classes, it also give a score of *how* negative or *how* positive a point is. We can use this to rank the points from most negative to most positive.

How to do this depends on the classifier. Here's how to do it for a linear classifier. We simply measure the distance to the decision boundary.We can now scale our classifier from timid to bold by moving the decision boundary from left to right.

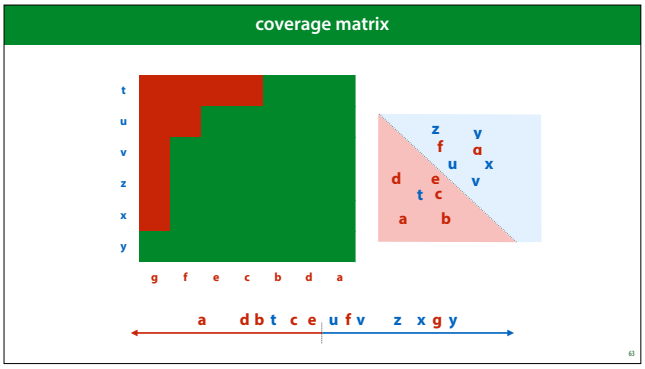
After we have a ranking, we can scale the eagerness of the classifier to make things positive. by moving the threshold (the dotted line) from left to right, the classifier becomes more eager to call things negative. This allows us to trade off the true positive rate and the false positive rate.



Now, we can't test a ranking on our test data, because we don't know what the correct ranking is. We don't get a correct ranking, just a correct *labeling*.

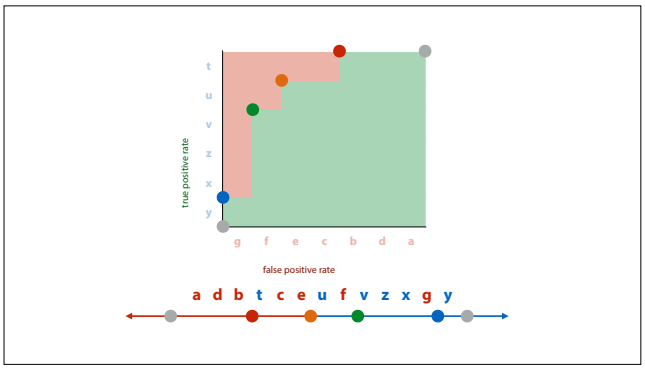
However, we can indicate for specific pairs that they are ranked the wrong way around: all pairs of different labels. For instance, **t** and **f** form a ranking error: **t** is ranked as more negative than **f**, even though **t** is positive and **f** is negative.

Note: a ranking error is a *pair* of instances that is ranked the wrong way around. A single instance can be part of multiple ranking errors.

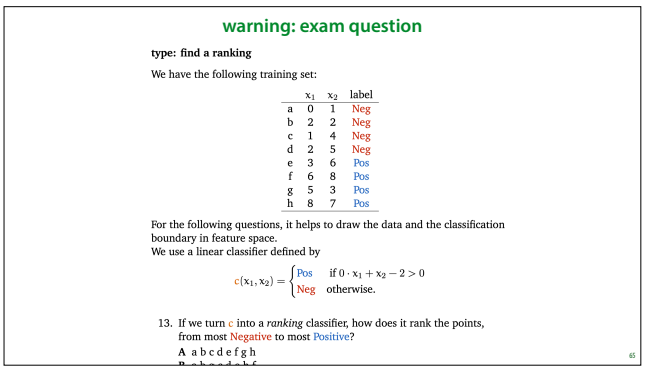


We can make a big matrix of all the pairs for which we know how they should be ranked: negative points on the horizontal axis, positive on the vertical. The more sure we are that a point is positive, the closer we put it to the bottom left corner. This is called a **coverage matrix**. We color a cell green if the corresponding points are ranked the right way round, and red if they are ranked the wrong way round.

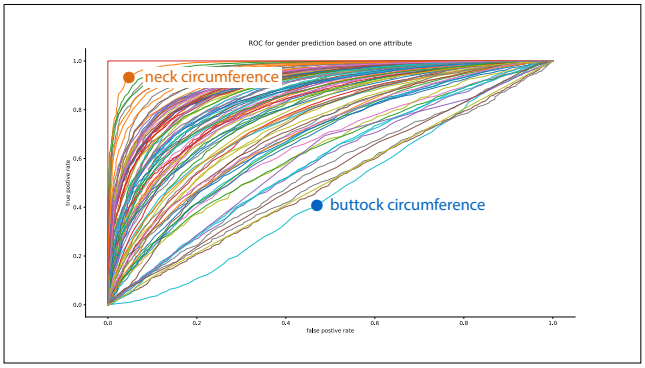
Note that the proportion of this table that is red, is the probability of making a ranking error.



The coverage matrix shows us exactly what happens to the true positive rate and the false positive rate if we move the threshold from the right to the left. We get exactly the kind of behaviour we talked about earlier. We move from the all-positive classifier step by step to the all-negative classifier.



This is one of the question types on the exam. There are more details in the third homework.

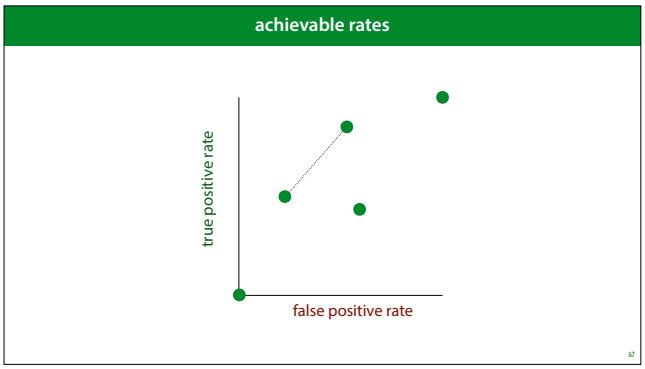


As an example of what real ROC curves look like, here are the ROC curves for predicting gender from every single physical measurement in the ANSUR II data used in the first lecture and in the second worksheet. The red curve, that makes a right angle is what we get when we use the target label itself as a feature.

The lowest AUC comes from using buttock circumference as a feature, and the high from using neck circumference. Remember that the male class are all soldiers in this dataset, so it's likely we're actually predicting whether somebody is a soldier more than their gender

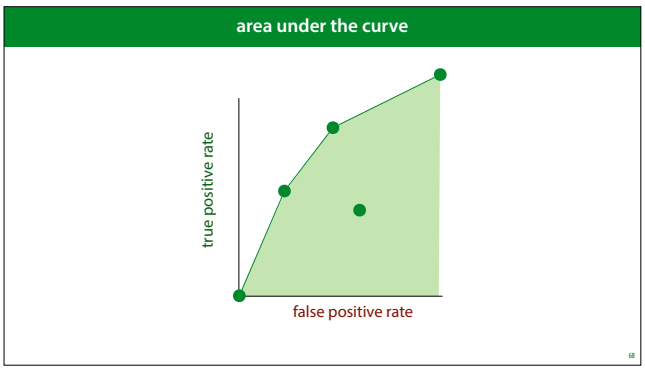
This plot is for the complete data so there is a 4:1 class

imbalance (male:female).



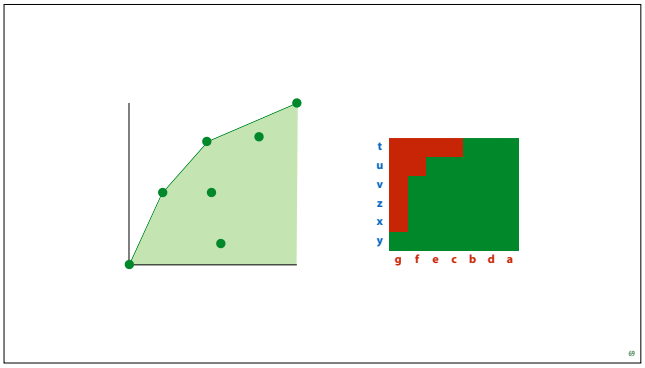
If we draw a line between two classifiers we know we can create, we can create a classifier for every point on that line simply by picking the output of one of the classifiers at random. If we pick with 50/50 probability we end up precisely halfway between the two.

If we vary the probability we can get closer to either classifier.



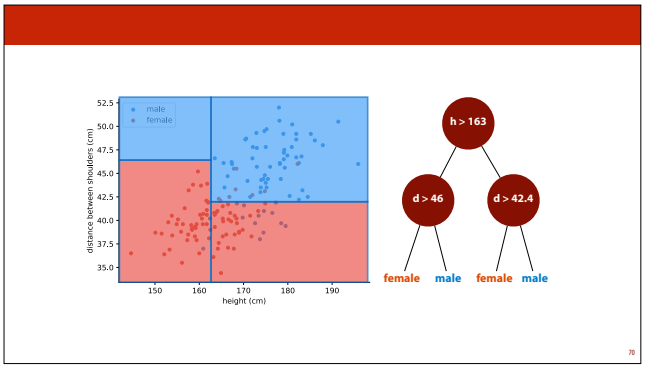
The area under the curve (AUC) of classifiers that we can create (the *convex hull* of all classifiers we've seen) is a good indication of the quality of the classifier. The bigger this area, the more useful classifiers we can achieve.

The AUC is a good way to compare classifiers with class or cost imbalance, where we don't know what our preferences are.



As we saw before: normalizing the coverage matrix gives us the ROC space (barring some small differences that disappear for large datasets). The area under the ROC curve is an estimate of the green proportion of the coverage matrix. This gives us a good way to interpret the AUC.

The AUC (in ROC space) is an estimate of the probability that a ranking classifier puts a randomly drawn pair of positive and negative examples in the correct order.

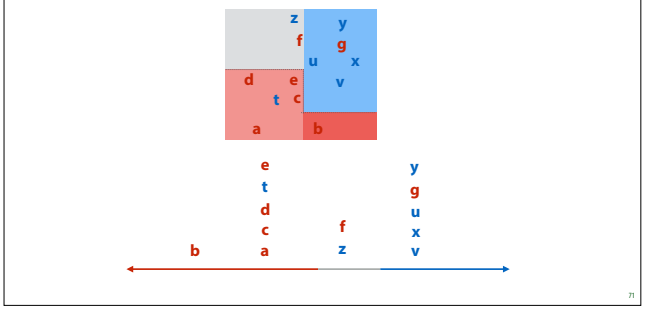


To re-iterate, **how we get a ranking from a classifier depends entirely on the model**. Let's look at one more example: the tree classifier that we saw earlier. Again, we'll discuss the actual algorithm for training decision trees later. For now, we'll just

The decision tree is an example of a *partitioning* classifier. Is splits the feature space into partitions, and assigns each partition, also known as a **segment**, a class. All instances in the segment get the same class.

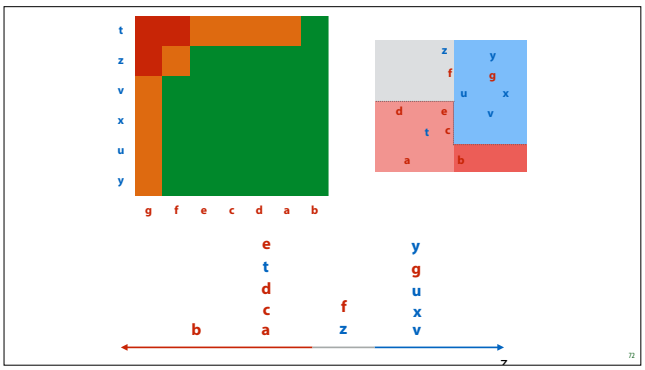
---

ranking decision tree



In this example we have an instance space that has been split into four segments by a decision tree. We rank the segments by the proportion of positive points. We then put all points in one region on the same level in the ranking.

In this example, **b** is more negative than **a**, because **b**'s segment contains only negative examples, whereas **a**'s segment contains a mix of positive and negative examples.



This means that for some pairs (like **f**,**z**), the classifier ranks them as "the same". We'll color these **orange**.

For large datasets, these regions will not contribute much to the total area under the curve.

---

important points

The confusion matrix and all metrics derived from it are metrics for a *single* classifier.

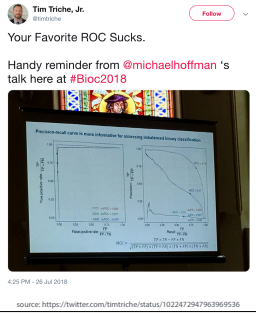
AUC is a metric for a *collection of classifiers*, usually derived from a ranking classifier.

How to turn a classifier into a ranking classifier, depends on the type of classifier.
For linear classifiers, take the distance to the decision boundary
For tree classifiers, sort by class proportion in each segment

AUC is a good metric if we don't know the relative importance of the classes, or if the classes are unbalanced.

To interpret the AUC, you should know not just what classifier was used, but how it was made into a collection of classifiers. You should also know whether it's a the area under an ROC curve, or a precision/recall curve.

ROC vs precision-recall



An alternative to the ROC is the **precision/recall curve**. It works in exactly the same way, but has precision and recall on the axes.

As you can see in this tweet, in many settings the PR curve can be much more informative, especially when you're a plotting the curves. Practically, it's little effort to just plot both, and judge which one is more informative.

ROC has the benefit of an intuitive interpretation for the AUC (the probability of ordering a random pair the right way round). I haven't yet found a similar interpretation for the PR-AUC.

---

setting the threshold

Show the user the ROC/PR curve, let them choose
This can be difficult to do accurately.

Estimate cost of misclassifications. Factor into the loss function. Minimize the expected cost.
In sklearn, this is done by setting *class weights*. If a false negative costs as much as three false positives, we set the positive weight to 3 and the negative weight to 1.

75

To put a classifier into production, a ranking may not be enough. Sometimes, you just need to produce a single answer. In that case, you can still use the ROC and PR curves to tune your hyperparameters and choose your model, but ultimately, you'll need to choose a threshold as well: the point at which you decide to call something a positive.

This is more of a software development issue than a scientific choice. Often, you have to look carefully at the curves, perhaps together with the end users, to make a decision.

The second approach works best with probabilistic classifiers, which we'll discuss next week.

---

recap

split your data into train/val/test

accuracy is great, unless you have class imbalance or cost imbalance

if you do, look at your:
- confusion matrix
- precision/recall space
- ROC space

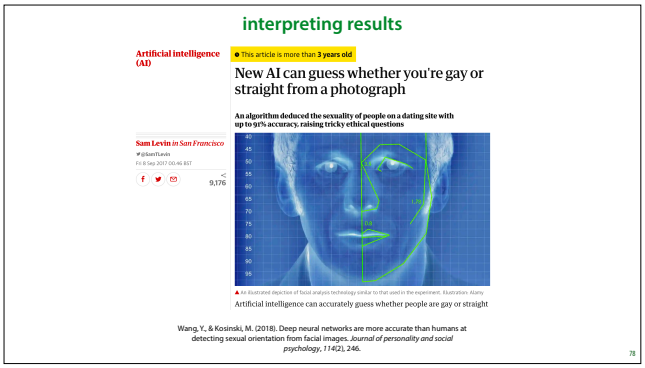
if you need a single number: try ROC-AUC or PR-AUC

76

Model Evaluation
Part 4: Social Impact 2

Machine Learning
mlvu.github.io
Vrije Universiteit Amsterdam

Model evaluation is not just about showing how well your model works. It's also about working out what it *means* to get a certain performance. And more importantly, what it doesn't mean.

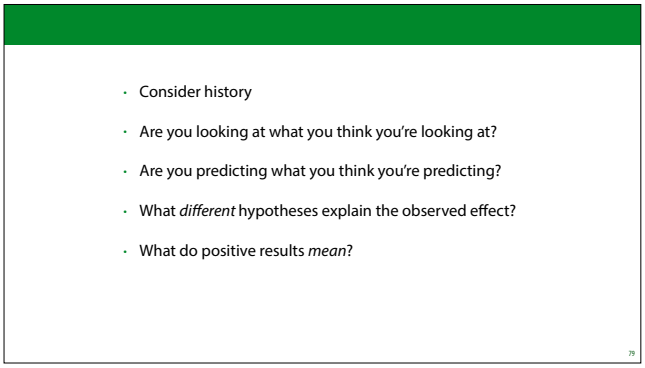


In this video, we will use the following research as a running example. In 2017 researchers from Stanford built a classifier that predicted sexual orientation (restricted to the classes "heterosexual" and "gay") from profile image taken from a dating site. They reported 91% ROC-AUC on men and 83% ROC-AUC on women.

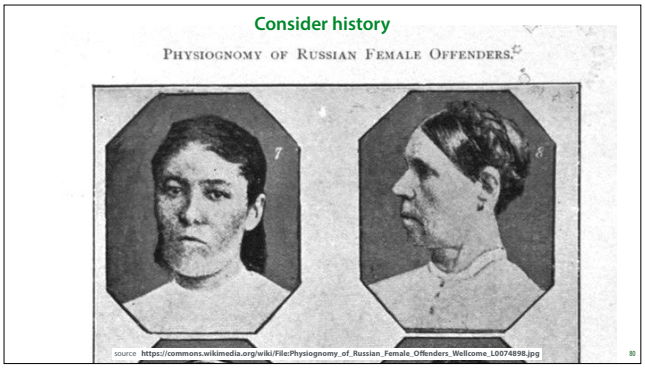
The results were immediately cited as evidence of a biological link between biology and sexual orientation. The following important caveats were largely overlooked in media reports:

- Some of theperformance came from facial landmarks (roundness, length of nose, distance between eyes, etc), but some came from superficial details like hairstyle, lighting and grooming. The accuracy
- The results were true when averaged over a large population. It's true that women live longer than men on average, but that doesn't mean that there are no old men. Likewise, the fact that you can guess orientation based on, say, the length of the nose, with better than chance accuracy, may only be due to a very small difference between the two distributions, with plenty of overlap.
- ~90% ROC-AUC may sound impressive, but it basically means that you will make 1 ranking error for every 10 attempts.

The study authors make may of these points themselves, but that didn't stop the paper from being wildly misrepresented: **https://docs.google.com/document/d/11oGZ1Ke3wK9E3BtOFfGfUQuuaSMR8AO2WfWH3aVke6U/edit#**



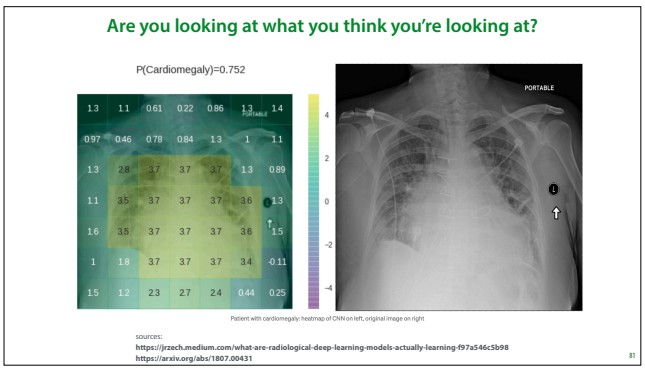
Like in the previous video, we'll look at some important questions to ask yourself when you come up against a topic like this. Let's ask the same questions again (with some new ones thrown in for good measure).



In these cases, it's important to know your history. Physiognomy is the study which attempts to infer character from facial features.

In this case there's is a long history of scientists claiming to be able to divine personal attributes (most often "criminality") from the structure of a subject's face. This is called physiogmony and almost any claim made has been conclusively disproven as false, and based on poor scientific practice and spurious correlations.

That doesn't mean, of course, that the entire idea of physiognomy is conclusively disproven. But is does mean that when we are stumbling into the same area with new tools, we should be aware of the mistake made in the past and, so that we can be careful not to repeat them.



The first thing to be aware of is what you're looking at. This is especially important with modern systems that can look at raw image data without extracting specific, interpretable features.

Here a visualization of a classifier looking at a chest x-ray and making a prediction of whether the patient has Cardiomegaly (an enlarged heart). The positive values in the heat map indicate that those regions are important for the current classification. The largest values are near the heart, which is what we expect.

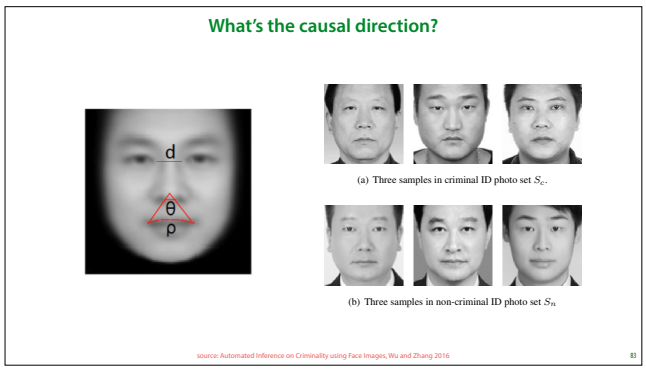
However, the classifier is also getting a positive contribution from the "PORTABLE" label in the top right corner and the marker on the right. These indicate that the x-ray was taken with a portable scanner. Such scanners are only used when a patient's condition has progressed so far that they can't leave their house. In such cases it's a safe bet that they have Cardiomegaly.

These problems are often called “Clever Hans” effects. Clever Hans, or der Kluge Hans, was an early-20th century German horse, who appeared to be able to do arithmetic.

As it turned out, Hans was not doing arithmetic, but just reading the body language of its handler, to see whether it was moving towards the right answer. This is impressive in itself, of course, but it does mean that Hans didn’t show the kind of intelligence that was being attributed to him.

Crucially, this was *not* a hoax. The handler truly believed that Hans was able to do arithmetic, and had no idea that he was guiding him subconsciously. This, incidentally is also why double-blind experiments are so important in other fields.

For us, Hans serves as a powerful reminder that just because we’re seeing the *performance* we were hoping for, doesn’t mean we’re seeing it for the *reasons* we were hoping for as well.



A related question you should ask when you find that you can successfully predict X from Y is **which causes which**?

The image on the left shows a feature that researchers found when attempting to predict criminality based on a dataset of faces of criminals and non-criminals. One of their findings is that the angle made by the corners of the mouth and the tip of the nose is a highly predictive feature. The authors suggest that such facial features are indicative of criminality

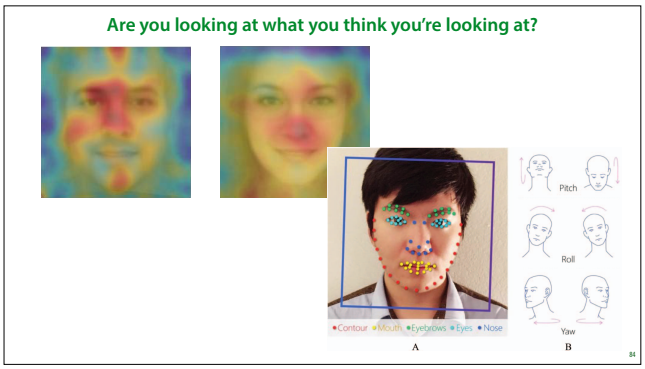
However, when we look at the dataset we see that it’s not the features of the face, so much as the expression that differs. In the “non-criminal” photographs, the subjects hold a light smile, as is common, whereas in the criminal set the expressions have a more explicitly relaxed jaw. What we’re seeing here are not facial *features*, so much as facial *expressions*.

This is important, because it changes the interpretation of the results completely. The physiognomical interpretation is that there is a biological mechanism that causes both criminality and a particular wideness of the mouth, and that this is determined at birth. The alternative explanation is that when people with a criminal background have their photographs taken, they are more likely to prefer a menacing expression than the average person is.

Note, incidentally, that the photos of criminals are not mugshots. The are described as “normal ID photos” by

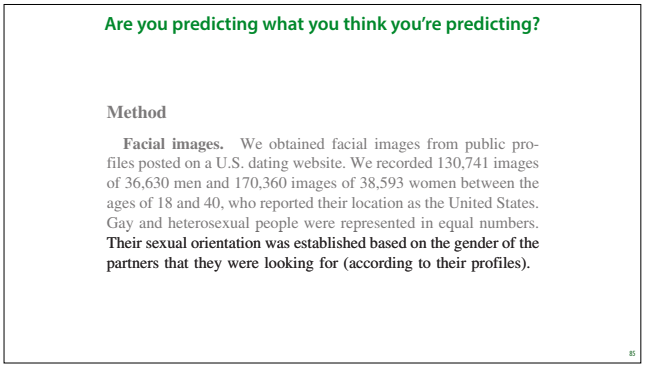
the authors.

Further discussion: **https://www.callingbullshit.org/case_studies/case_study_criminal_machine_learning.html**



Let's go back to the sexuality classifier. what might a Clever Hans effect look like here? In the most extreme case, you might expect a classifier just to look at the background of the image. The authors were more careful here than you might expect:

- The background of the image was blurred and the the facial features (eyes, nose, mouth) were detected and aligned.
- The focus of the classifier was investigated with saliency maps (indicators of where the model is looking). This is a fallible method, but it does show a general focus on the face.
- A second classifier was fed only facial landmarks: the position of the eyes, roundness of the jaw, etc. The suggestion being that this prevents Clever Hans effects.
- The deep neural network used to extract features from pixels was not trained on this data, but on another facial dataset. Only its *features* were fed to a shallow classifier that learned from these labels. This limits the ability of the classifier to pick up on surface detail.



Another question we suggested in the last video is whether the target label you've chosen is saying what you think it's saying.

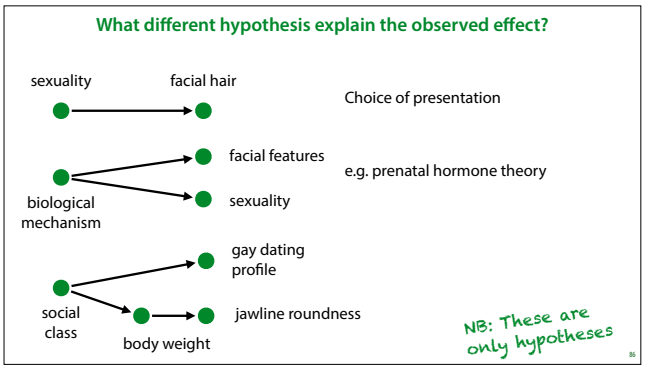
Here, the authors inferred sexuality from the stated preference in the dating profile. This is clearly correlated with sexuality, but not the same thing. Firstly, sexuality is one of those attributes (like movie genre) that can only be crudely approximated by a set of finite categories. Moreover, for many people it's not a fixed attribute, and it is subject to some evolution throughout their life.

The stated preference on a dating profile also means that you are capturing only those gay people who are

willing to live (relatively) openly as gay. This may be highly dependent on social background. It's certainly conceivable that in poorer subcultures, people are less likely to come out as gay, either to their community or to themselves.

This means that what we're detecting when we're classifying a face in this dataset as "gay" is more likely a combination of factors that are correlation to that label.

Incidentally, note the size of the dataset. One thing the authors can't be accused of is finding spurious correlations. It's a question of what the correlations that they found mean, but with this amount of data, as we saw before, we get very small confidence intervals,



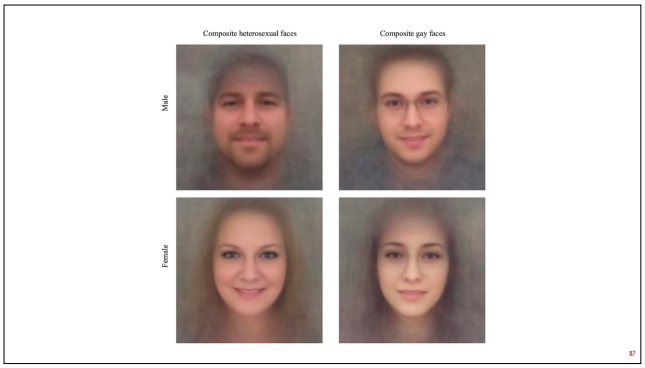
So, what kind of hypotheses can we think of for what is causing the performance of the classifier?

The authors observe that in their dataset the heterosexual men are more likely have facial hair. That's most likely to be a grooming choice, based on the differences in gay and heterosexual subcultures.

For other correlations, such as that between sexuality and nose length, the authors suggest the prenatal hormone theory, a theory that relates prenatal hormone levels in the mother with the sexuality of the subject. In short, a biological mechanism that is responsible for both the (slight) variation in facial features and the variation in sexual preference.

But that's not the only possibility. In the previous slide, we saw that it's difficult to separate facial features from facial expressions. However, even if we somehow eliminate the expression, that doesn't mean that every facial feature we see is determined at birth. For instance, the roundness of the jaw is also influenced by body weight, which is strongly influenced by social class (for instance, whether somebody grows up poor or rich). And while there's no evidence that social class influences the probability of *being gay*, it most likely does influence how likely a gay person is to end up setting up a dating profile.

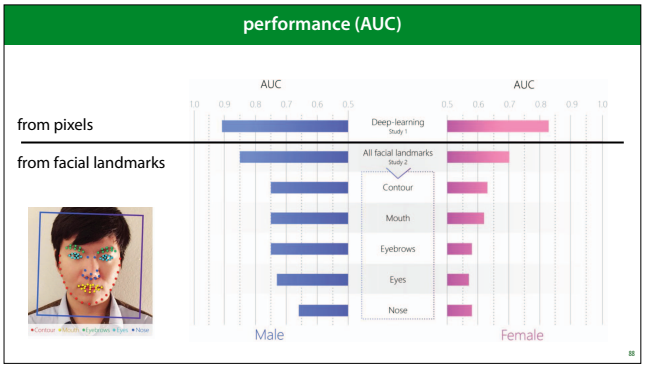
Note that these are purely hypotheses, intended to show which kinds of causalities can cause these correlations. I'm not in the least bit qualified to say which is more likely to be true.



The authors plotted the four averages faces for the classes male/female and gay/heterosexual in their dataset. Here are the four options. It's a peculiar property of datasets of (aligned) faces that the mean is often quite a realistic face itself.

Consider this plot with the hypotheses on the previous slide. What differences do you see? Pay particular attention to the differences in skin tone, grooming, body weight, and the presence of glasses.

I'll leave you to decide which you think is the more likely explanation for these difference: choice of presentation, social class, or sexuality.



The authors repeated their experiments by classifying based purely on facial landmarks. The idea is that we can detect landmarks very accurately, and classifying on these alone removes a lot of sources for potential Clever Hans effects.

We see that all subsets of the face landmarks allow for some predictive performance, but there is a clear difference between them. Note that just because we are isolating landmarks, doesn't mean that we are focusing only on biological causes. As we saw earlier, the shape of the mouth is determined more by expression than by facial features, and the roundness of the jaw is partly determined by body weight, which is correlated with social class.

We will discuss the AUC metric in the third lecture. For now, you can think of a classifier with 81% AUC as one that, given a random pair of gay and heterosexual instances from the data, will successfully select the gay instance 81% of the time.

Let's assume that the shape of the nose is mostly unaffected by grooming and expression. I have no idea whether this assumption is valid, but say that it is. Focusing purely on the shape of the nose, we see that performance drops to 0.65 AUC for men and 0.56 AUC for women. This is still better than chance level.

Can we say that homosexuality can be *detected* based on the shape of the nose? Can we conclude a biological relation based on this correlation?

What does 0.56, 0.65 AUC mean?

Guessing sex or gender based on:	Accuracy proportion of incorrect classifications	AUC chance of correctly ordering a random male/female pair
Age Dutch census data	0.51	0.52
Age for people aged 80 and over Dutch census data	0.61	0.57
Age ANSUR II data	0.65	0.59
Waist circumference ANSUR II data	0.68	0.74
Stature (height) ANSUR II data	0.84	0.92

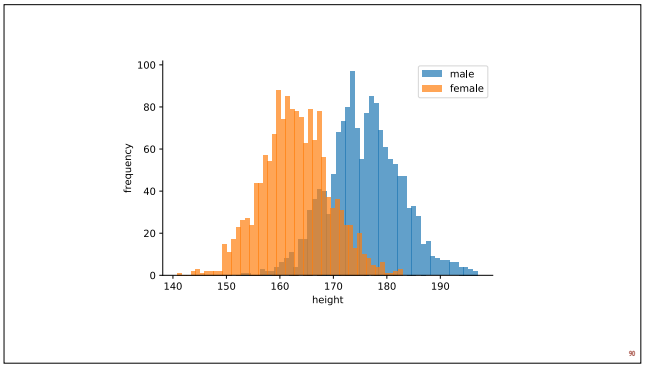
To interpret numbers like these, it's good to get some points of reference. If we try to guess somebody's gender or sex (the distinction doesn't matter much for such a crude guess), knowing nothing about them except their age, the best be can expect to do is slightly better than chance level (51% of our guesses will be correct). The reason we can get better than chance level is that women tend to live longer than men. This means that if we guess "female" for older people, we are a little bit more likely to be correct.

If we restrict ourselves to older people, the effect becomes more pronounced and we can get an accuracy as high as the sexuality classifier got based purely on noses in the female part of the data. This can help us to

interpret the AUC the authors achieved. Note that this accuracy is achieved by calling everybody female, and the AUC is achieved by guessing that the older person in a pair is always female. Think about that. **If you walk into a carehome blindfolded and simply call everybody female, can you really claim to be *detecting* their gender?**

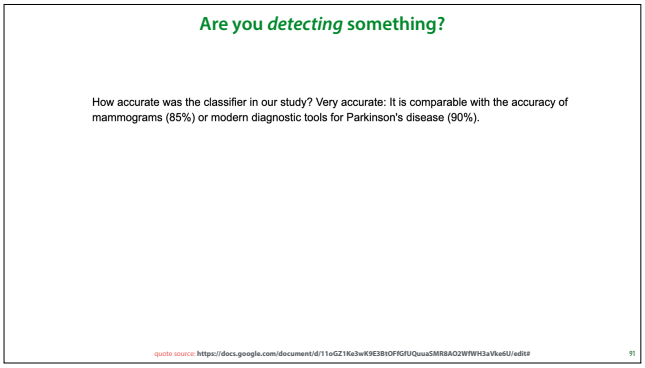
If we look purely at people's height, we get an accuracy and AUC that is comparable to what the authors achieved from the pixel data. This is also an important point to consider. Height and gender are correlated, but that doesn't mean that there are no tall women or short men. It also doesn't make tall women "masculine", or short men "feminine". It's just a slight correlation that allows us to make an educated guess for certain parts of the range of heights.

NB: In the ANSUR II data, the subjects are soldiers. It's possible that some sex differences are more pronounced in this population due to selection effects or physical training. We balanced the ANSUR data by subsampling to make the numbers of male and female subjects equal.



Here are the histograms per gender for the ANSUR data. There is a big discrepancy, but notice also how big the area of overlap is.

This is always what we should imagine when people say that property A is predictive for attribute B. Just because there's some difference between the populations doesn't mean that there are no short men or tall women. And most importantly it doesn't mean that being short makes you in some way more feminine or being tall makes you in some way more masculine.



Are you really *detecting* gender when your call somebody male just because they're tall? The authors compare their classifiers to medical diagnostic tools to provide an interpretation of the AUC scores.

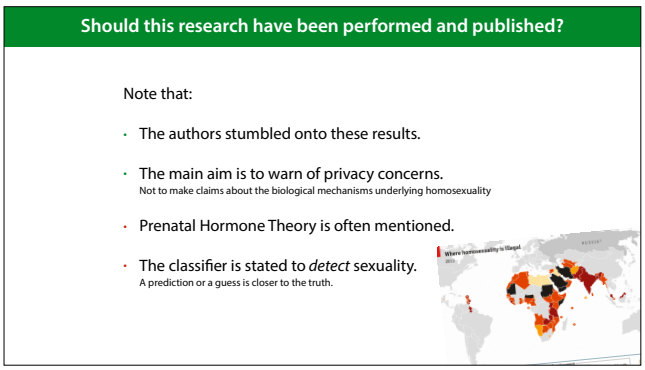
This is where we must make a clear distinction between what a classifier like this does and what a diagnostic test does. A test like that for breast cancer looks explicitly only at one particular source of information. In this case the mammogram. The clinician will likely take the result of this test, and factor in contextual clues like age and lifestyle if the test is unclear. The test can be said to **detect** something, because it is strictly confined to look at only one thing.

The clinician is **predicting** or **guessing** something based on different factors. One of which is the test.

The diagnosis of Parkinson's is similar to the way this classifier works, there is no unambiguous diagnostic tool like a blood test, so the diagnostician needs to look at contextual clues like medical history, age and risk factors.

There is still a difference, however, in that the features are made more explicit. The pixel-based sexuality classifier may be inferring social class from the image, but it's not telling us that it's doing this. A doctor may be guilty of such subconscious inferences as well, but we can expect a greater level of *interpretability* from them.

NB: The authors use the word accuracy to refer to ROC-AUC.



After all that, it's natural to ask whether this research project was a mistake. In short, were the authors wrong to do this, and if so in what way. This is a question of values rather than science, so it's up to you to decide. I'll just note some important points to consider.

Firstly, the authors weren't looking to prove this point one way or another, and initially stumbled on to this result. Given that a result has been established, it's most often unethical *not* to report it. So long as that reporting is done carefully and responsibly, of course.

The stated aim is not to make any claims about causal mechanisms. The authors are less interested in whether the classifier picks up on grooming choices of biological features, than in whether the guess can be made with some success at all.

What we may blame them for, is poor framing. The use of the word detecting is subject to misinterpretation. To be fair, that's something I only became aware of when looking into this matter, so I'm not sure I can be very judgemental in that respect.

Another odd thing, is that in both the paper and the explanatory notes, prenatal hormone theory is often mentioned. The experiments shown here provide no evidence for one causal hypothesis over another, so it would probably have been better to make no claims about causal effect whatsoever.

How do you frame your research?

Consider which features you are using.

Consider multiple hypotheses. Social, biological, personal.
Train yourself to always come up with different explanations for a given set of facts.

Distinguish between **detecting**, **predicting** and **guessing**.
Even 0.91 AUC is more guessing than predicting. It's only detecting if you strictly control your features.

93

acknowledgements

Thanks to Jasmijn Bastings (@BastingsJasmijn) and Anne Ogborn (@AnneOgborn) and others for discussing this topic with me over Twitter.

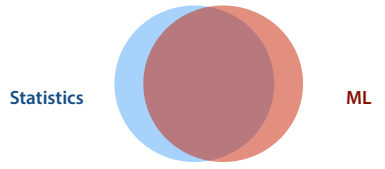
94

I'm indebted to people on twitter for helping me to see the different angles on this topic.

Model Evaluation
Part 5: Statistical Testing

Machine Learning
mlvu.github.io
Vrije Universiteit Amsterdam

Machine Learning vs. Statistics



Stats but not ML: Analyzing research results. Experiment design. Courtroom evidence.
More ML than Stats: Spam classification, movie recommendation,.

96

As noted in the first lecture, statistics an ML are very closely related. It's surprising, then that when we perform ML experiments, we use relatively little of the statistics toolkit. We don't often do **significance tests**, for instance.

should we do statistical tests at all?

- Makes ML experimentation difficult. Lots of disagreement.
- People overestimate the value of statistical analyses.
- Does not promote the best methods
- The ultimate validation of research is REPLICATION

On the appropriateness of statistical tests in machine learning, Janez Demšar, 2008
Machine Learning as an Experimental Science (Revisited), Chris Drummond, 2006

97

Note everybody agrees. Hypothesis testing comes with a lot of downsides. Given that we usually have very big sample sizes (10000 instances in the test set), our efforts may be better spent elsewhere.

Another consideration is that the ultimate validation of research is replication, not statistical significance. Somebody else should repeat your research and get the same results. Because all of our experimentation is computer code, a basic replication could be as simple as downloading and running a docker image. After that it's easy to try the same on new data, or check the model for bugs.

Since the community is so divided on the question, we won't emphasise reporting statistics on experimental results too much for this course. However, there are a few points worth mentioning.

---

statistical analysis for reported results

Can the observed results be attributed to *real characteristics* of the models under scrutiny or are they observed *by chance*?

98

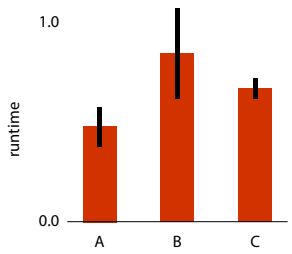
This is the main question that statistical analysis is meant to answer.

When it comes to reporting results, the question arises whether we should report our results with some sort of statistics. If we report that classifier A is better than classifier B because their accuracies are .997 and .998 respectively, can we really trust that statement? We've seen how much difference a little bit of noise can make, shouldn't we be reporting some kind of hypothesis test on that statement, to show that we've used enough test data?

quote source: **http://www.icmla-conference.org/**

---

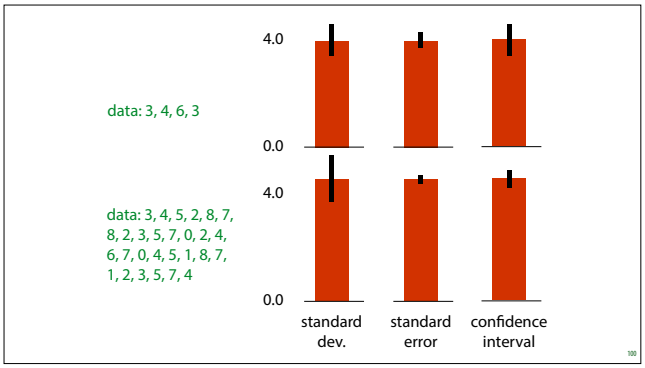
error bars



99

First, error bars.

If you see a picture like this, showing the mean runtime of an experiment, measured for three models, and averaged over a number of runs, what would you imagine the error bars denote?

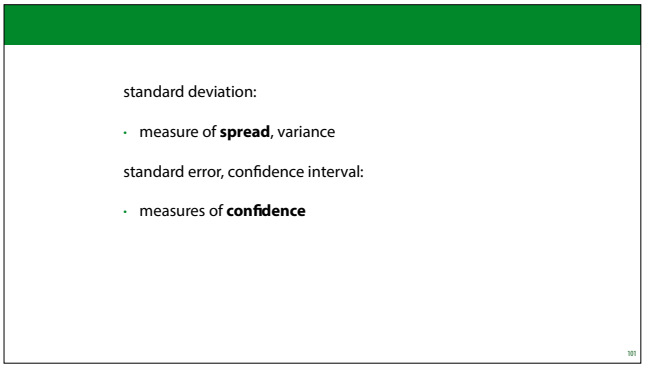


The truth is, that there is no standard definition for what error bars denote, and if the authors didn't specify what their error bars indicate, the authors messed up.

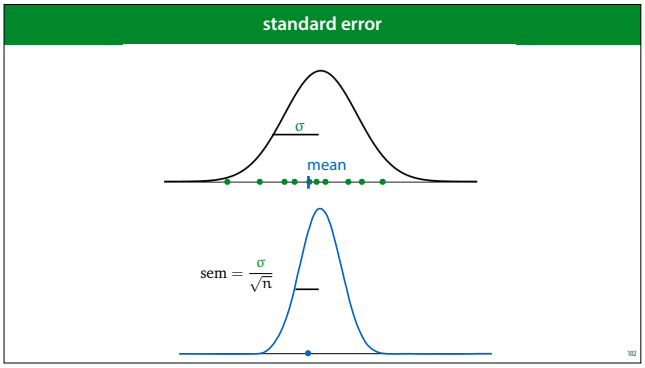
These are the three most common options. Before we explain exactly what they mean, let's look at how they behave, specifically when we increase the amount of data.

If we sample more data, the estimate of our **standard deviation** becomes *more accurate*. It's an estimate of a property of our data distribution.

The standard error and the confidence interval are indicators of how confident we are about our estimate of the mean (indicated by the height of the error bar). There more data we have, *the smaller they get*.



We'll assume you know what the standard deviation of a dataset is.

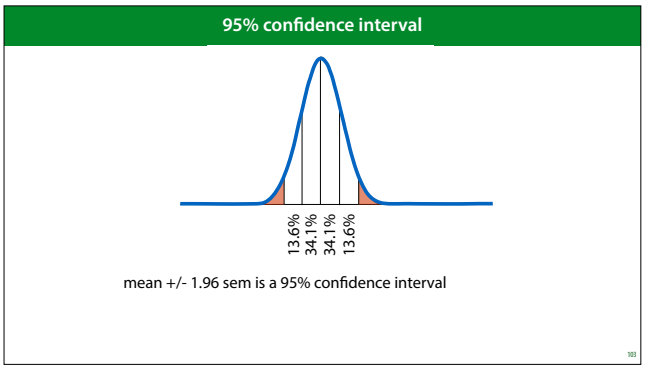


Here is how the standard error (SEM) works. If we sample some data (like error values) and calculate their mean, the result is random value too. If we resample, we get a (slightly) different mean. This means we can plot the probability distribution of this value.

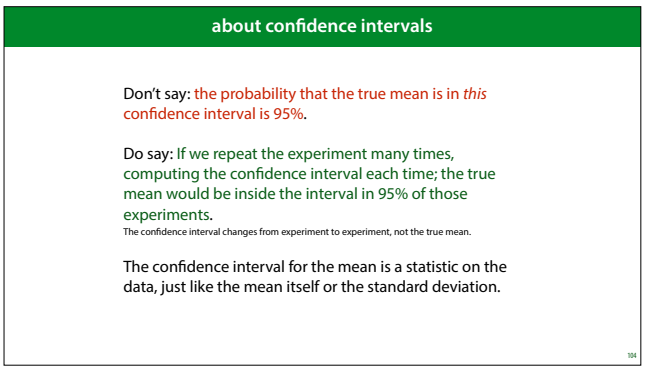
If our original distribution is normal with standard deviation sigma, the distribution of the sample mean is approximately normal too (for large enough samples), with standard deviation sigma divided by the square of the number of samples.

The original standard deviation is not usually known, so it is estimated from the data. The bottom

distribution is a Student's-t distribution. For large enough sample sizes (more than 100), we can treat this as a normal distribution.

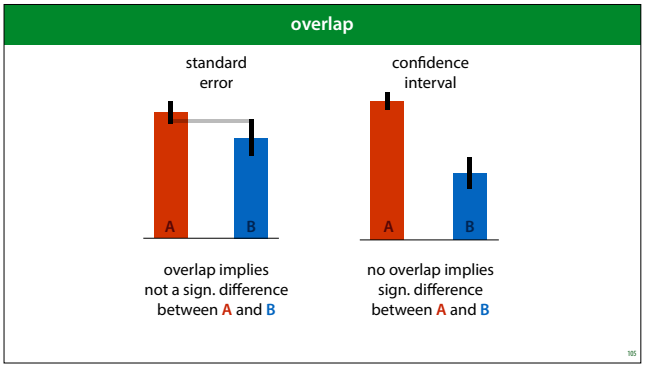


As you may know, the region of four standard deviations around the mean of a normal distribution contains roughly 95% of the probability mass. This is what the confidence interval error bar is: it is simply twice the standard error on both sides of our estimate, and it gives us a region for which we are 95% confident that it contains the true mean.



This is an important distinction. These are frequentist methods, so there is no probability associated with the true mean at all. It is simply an objective, determined value (which we don't know). The probability comes from sampling, and from computing the interval from a sample.

So instead of having a fixed interval, with the true mean jumping around probabilistically, we have a fixed true mean around which we get an interval that jumps around if we resample the data The probability of it jumping so much that it no longer contains the true mean is 5%.

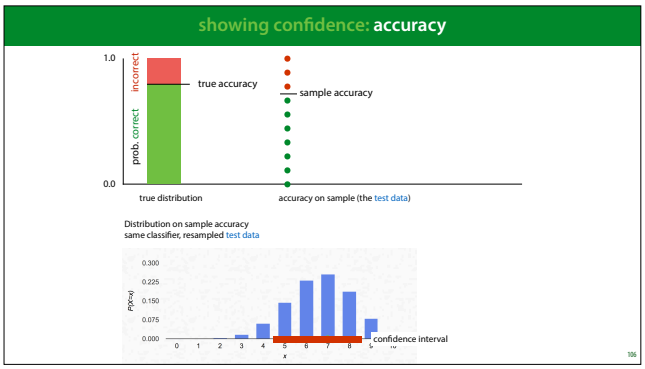


Say you plot the mean squared error for regression models **A** and **B**, together with some error bars. Does the the fact that the error bars overlap or not tell you whether the measured difference between the two models is statistically significant?

Yes, for standard error bars, the existence of overlap implies that there is no significant difference between the two effects (i.e. the possibility that the difference is due to random chance is high, and a repeat of the experiment on new data may show a different result). If you plot confidence interval error bars, and there is no overlap, you may conclude that the difference between the models is significant. If you repret the experiment on fresh data, it is very likely that model **A**

would beat model **B** again.

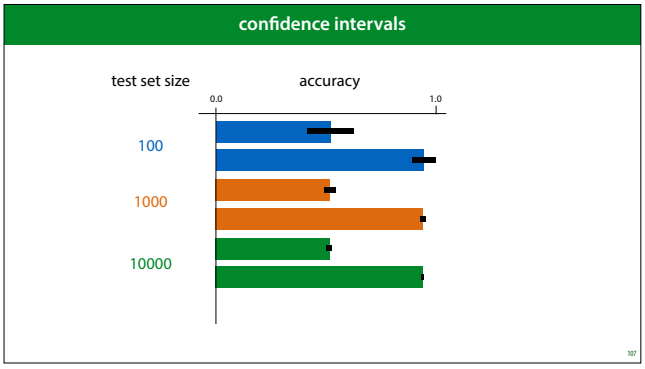
In both cases, the converse does not hold. If the SEM error bars do not overlap, there may or may not be a significant difference. If the confidence interval error bars do overlap, there may still be a significant difference, depending on how much they overlap.



The true accuracy of a classifier (the probability of a correct classification) is also one of these "true values", which we can't see, but that we estimate by a sample mean, where the sample is our test set. The accuracy that we actually see is an estimate of a true value. Each instance in our test set is like a flip of an unfair coin that lands heads for a correct classification and tails for an incorrect one. This is called a *Bernoulli* distribution.

Since this is a random process (our test set is randomly sampled), we can work out the distribution over the outcome (the number of correct classifications). The number of positive outcomes over a fixed-size sample from a Bernoulli distribution is described by a Binomial distribution. A region around our sample of 7 successes that contains 95% of the probability mass is a 95% confidence interval for our estimate of the accuracy.

This is the same process as shown on slide 67, but with a Bernoulli distribution instead of a Normal one.
source: **https://homepage.divms.uiowa.edu/~mbognar/applets/bin.html**



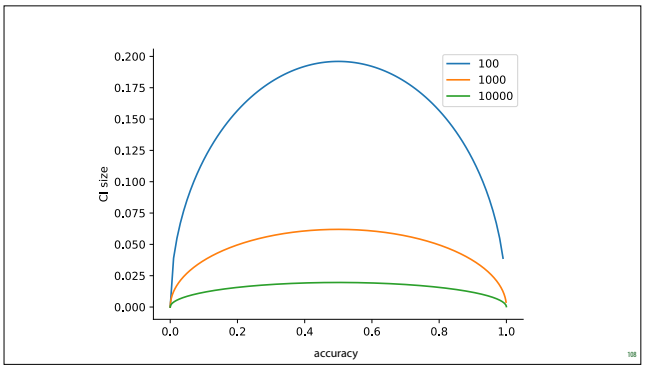
The size of this confidence interval depends on two factors: the true accuracy* and the size of the test set. Here are some examples for different accuracies and test set sizes.

This tells us that if the true success probability (accuracy) of a classifier is 0.5, and the test set contains 100 examples, our confidence interval has size 0.2. This means that even if we report 0.5 as the accuracy, we may well be wrong by as much as 0.1 either side.

Even if these confidence intervals are usually not reported, you can easily work them out (or look them up) yourself. So, if you see someone say that classifier A is better than classifier B because A

scored 60% accuracy and and B score 59%, on a test set of 100 instances, you have reason to be sceptical.

We don't know the true accuracy, but it's accepted practice to assume that the estimate is close enough to get a reasonable estimate of the confidence interval.



Here are the full curves, in case you every need to look it up.

Confidence depends on *the size of the test set*.

Avoid small test sets.

If you can't, look into Alpaydin's 5x2 F test
https://www.cmpe.boun.edu.tr/~ethem/files/papers/NC110804.PDF

The main takeaway here is that confidence intervals are usually not necessary if your test set is big enough. With a test set of 10 000 instances, you can comfortably tell the difference between 98% and 99% accuracy without having to worry about random effects.

If you don't have the luxury of a large test set, you may need to do some statistical testing to see whether the effect you've observed (classifier A is better than classifier B) is genuine or down to random chance. It's generally accepted that Alpaydin's 5x2 cross validation is the best test for this purpose. It's out of scope for this course, but follow the link if you run into this problem.

---

why use statistics in ML

- to show *confidence*
- to show *spread*

Confidently show the performance of the best model you found, and then measure the variance of the method you used to find it.

Showing confidence means showing how reliable our numbers are. Standard error and confidence intervals are good ways to show confidence (or lack thereof).

Showing spread is more about providing insight to the user. Say I train a classifier by gradient descent. If I have a big test set, I can very *confidently* measure and report the accuracy of this particular classifier. However, gradient descent uses *random* initialization. If I repeat the training process, I may end up in a different local minimum, and get a different classification performance. It's likely that I also want to communicate how much the measured performance is dependent on this randomness.

showing spread

Sources of randomness:

- Data sampling
- Search algorithm (i.e. initializing gradient descent)

Report standard deviation, describe *what* you repeat.

- How do you repeat data sampling?

If we have a large enough test set, we know that the confidence interval is small enough. But we do want to know how much the randomness in our process affects the result. What is the probability that repeating the process (on the same data, or on new data) produces wildly different results?

For factors like the initialisation of gradient descent, this is easy to test: you just rerun a few times on the same data. But how do you test how robust the result are against sampling a new dataset?

---

resampling

Cross validation again, on the whole data set.

Stratified cross-validation (keeps the class proportions the same in all folds)

Leave-one-out cross-validation, a.k.a. the jackknife method

Slight bias: smaller datasets.

---

bootstrapping

Sample, with replacement, a dataset of the same size as the whole dataset.
On average, about 63.2% of the dataset will be included. The rest will be duplicated instances.

Each bootstrapped sample lets you repeat your experiment.

Note that some classifiers will respond poorly to presence of duplicate instances.

Better than cross validation for small datasets.

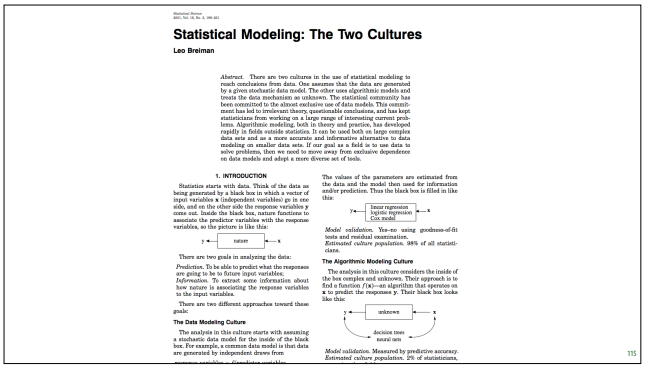
---

statistics: summary

Don't worry too much about it (until you have to).
Even in top ML conferences, rigorous statistical analysis is relatively rare.

Distinguish between showing *confidence*, and showing *spread*.

Think about what you want to claim, and what analysis would make your claim as strong as possible.



Statistical Modeling: The Two Cultures

Leo Breiman

Abstract. There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown. The statistical community has been committed to the almost exclusive use of data models. This commitment has led to irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems. Algorithmic modeling, both in theory and practice, has developed rapidly in fields outside statistics. It can be used both on large complex data sets and as a more accurate and informative alternative to data modeling on smaller data sets. If our goal as a field is to use data to solve problems, then we need to move away from exclusive dependence on data models and adopt a more diverse set of tools.

1. INTRODUCTION

Statistics starts with data. Think of the data as being generated by a black box in which a vector of input variables **x** (independent variables) go in one side, and on the other side the response variables **y** come out. Inside the black box, nature functions to associate the predictor variables with the response variables, so the picture is like this:

There are two goals in analyzing the data:

Prediction. To be able to predict what the responses are going to be to future input variables;

Information. To extract some information about how nature is associating the response variables to the input variables.

There are two different approaches toward these goals:

The Data Modeling Culture

The analysis in this culture starts with assuming a stochastic data model for the inside of the black box. For example, a common data model is that data are generated by independent draws from

The values of the parameters are estimated from the data and the model then used for information and/or prediction. Thus the black box is filled in like this:

Model validation. Yes–no using goodness-of-fit tests and residual examination.

Estimated culture population. 98% of all statisticians.

The Algorithmic Modeling Culture

The analysis in this culture considers the inside of the box complex and unknown. Their approach is to find a function $f(\mathbf{x})$—an algorithm that operates on **x** to predict the responses **y**. Their black box looks like this:

Model validation. Measured by predictive accuracy.

Estimated culture population. 2% of statisticians,

115

If you're interested in the difference between machine learning and statistics, I can recommend this paper by Leo Breiman. It shows the difference between the two cultures. It makes clear that the machine learning approach of measuring models purely for predictive accuracy on a large test set, has a lot of benefits and makes the business of statistics a lot simpler and more effective.

Model Evaluation

Part 6: No Free Lunch

Machine Learning
mlvu.github.io
Vrije Universiteit Amsterdam

We end with a short, but important discussion.

A question that often arises is that of which classifier, model, search method, etc. is the best, independent of the data. Before we see the data, can we make a best guess for which approaches to try? Are there some methods that always work?

The no-free-lunch theorem(s)

Wolpert & MacReady 1997

"… any two optimization algorithms are equivalent when their performance is averaged across all possible problems"

source: Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67-82.

118

Given some data **X** and basic methods **A** and **B**?

Meta-methods:

- **method C**: Use a data split, choose whichever performs the best.
- **method D**: Use a data split, choose whichever performs the **worst**.

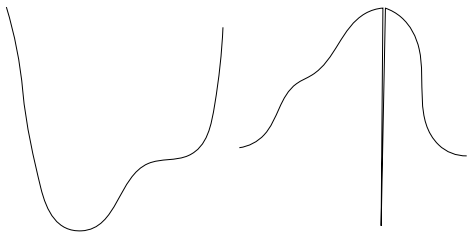
According to the NFL theorem, there are as many datasets X for which **C** beats **D** as there are for which **D** beats **C**.

119

Note that, intuitively, method D would be an absolutely insane method to choose a model.

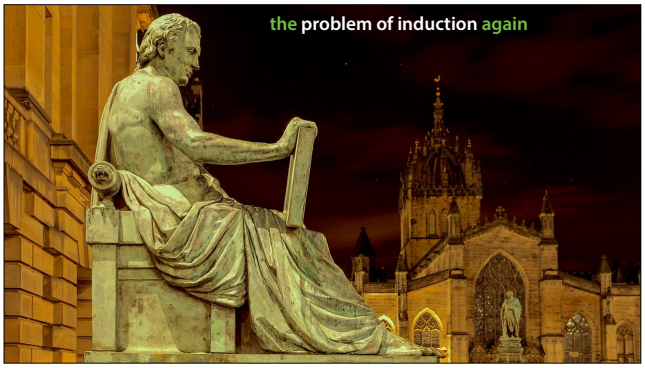
---

gradient descent



120

Here is another example. We know that gradient descent works well: to find the lowest point on a loss surface, you just follow the curve of the loss surface downward. However, for every loss surface and which gradient descent works, we can create a loss surface (like the one on the right) for which gradient ascent fails miserably, and actually, the opposite strategy works better: you have to climb to find the lowest point.



So we're back to the problem of induction. For any given situation where a learning method works, there's a situation where it doesn't.

And we can't tell which situation we're in algorithmically, because then we could use that and beat the NFL theorem.

We need *some* assumptions about the nature of the universe to understand why learning works at all.

---

the universal distribution

Not all datasets are created equal. The datasets for which our method works, are the likely ones.

The universe "generates" data for which our methods work

- Compressible data
- Simple data

The datasets that don't work aren't selected, because they look random to us.

We only understand those parts of the universe that generate understandable data

122

One "out" to the NFL Theorem, is that there is a "universal distribution" governing the data gathering process. The NFL Theorem implicitly assumes that all datasets are equally likely. Since this is not the case, we can work out a universally best algorithm (under the universal distribution).

Occam's razor

"The simplest explanation is often the best"

We should bias our algorithms towards simple models.

- Reduces overfitting, helps generalization
- Why should this be the case? (Domingos, 1999)

123

the no-free-lunch *principle*

There is no single best learning method. Whether an algorithm is good, depends on the domain.

124

Whether or not the NFL theorem means anything for us in practice, it has also given rise to a general *principle*, commonly followed in machine learning practice. The principle is that we should choose our method to deal with the task at hand, and not look for a universally best method.

Note that this is distinct from the NFL theorem, because everybody still uses data splitting universally to evaluate *which of these many methods* is the best. And by the NFL theorem, model selection by data splitting is also not a universal algorithm.

inductive bias

The aspects of a learning algorithm, which implicitly or explicitly make it suitable for certain learning problems and unsuitable for others.

A linear method has an *inductive bias* for linear relations.

125

This is an increasingly important phrase in machine learning. The business of the ML researcher is create models with helpful inductive biases. The business of the data scientist is to figure out what inductive bias might be helpful for the task at hand.